UQAC
Université du Québec
à Chicoutimi

MÉMOIRE

PRÉSENTÉ À

L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

MOHAMMED BENAICHA

IDENTIFICATION DES CONCEPTS POUR LA RÉ-INGÉNIERIE DES ONTOLOGIES

AVRIL 2017

RÉSUMÉ

Des services élémentaires telles que la restructuration (Ang. refactoring), la fusion (Ang. merge), l'extraction de modules (Ang. modularization), etc., sont indispensables pour la mise en oeuvre d'une plateforme de génie ontologique dont l'un des objectifs essentiels est d'assurer la qualité d'une ontologie qui risque de se détériorer avec l'usage.

INUKHUK est une plateforme de ré-ingénierie d'ontologies dont les services ci-haut sont basés sur un cadre formel, dit *Analyse Relationnelle de Concepts* (ARC). L'ARC a le pouvoir de créer de nouvelles abstractions sur n'importe quel type d'éléments ontologiques (concept, propriété, etc.). Ces abstractions factorisent des descriptions communes à plusieurs éléments et peuvent servir à corriger et/ou enrichir l'ontologie de façon à augmenter sa qualité. Toutefois, ces abstractions, qui sont le fruit d'un calcul mathématique, sont dépourvues de toute sémantique et par conséquent nécessitent un effort de conceptualisation afin d'être justifiées et intégrées à une ontologie. Le but de ce travail est de fouiller le bien fondé d'une abstraction et par la suite l'annoter avec un nom (En Anglais *label*) de concept. Les retombées sont multiples. D'abord, l'ontologie restructurée est beaucoup plus compréhensible par les experts et utilisateurs car l'enrichissement est autant structurel que sémantique. Ensuite, l'application des métriques de qualité basées sur l'analyse du vocabulaire ontologique pour l'estimation de la qualité de l'ontologie restructurée (*Ang.*, refactored) est tout à fait justifiable.

Pour ce faire, plusieurs méthodes de fouille ont été envisagées. La première méthode consiste à l'identification d'un nouveau concept ontologique à partir de la description de l'abstraction générée par l'ARC. La deuxième méthode consiste à confronter l'abstraction générée par l'ARC à des ressources linguistiques et/ou ontologiques existantes telles que WORDNET, structure catégorique de WIKIPEDIA, DBPEDIA, etc.

Les deux approches ci-haut ont été implémentées au sein d'un outil, dit TOPICMINER, qui fait désormais partie de la plateforme INUKHUK. TOPICMINER a fait l'objet de plusieurs expérimentations et a retourné des résultats satisfaisants selon le protocole de validation mis en place.

REMERCIEMENTS

Tout au long des différentes phases de mon présent mémoire, j'ai été soutenu par un bon nombre de personnes. Mes remerciements s'adressent en particulier à :

— Mon directeur de recherche, Monsieur Abdenour Bouzouane, professeur au département d'informatique et mathématique de l'université UQAC. Je tiens à le remercier vivement pour avoir accepter de diriger mes recherches, pour ses innombrables conseils, son suivi avisé et finalement pour tous ses efforts dans la lecture et la correction de ce mémoire.
— Mon co-directeur de recherche, Monsieur Roger Nkambou, professeur au département d'informatique à l'université UQAM et directeur du laboratoire de recherche en gestion, diffusion et acquisition des connaissances (GDAC). Je lui exprime mes sincères remerciements pour m'avoir pris en charge dans son laboratoire, pour son soutien scientifique et méthodologique et pour le temps qu'il a consacré à la correction de ce document.
— Monsieur Amine Mohamed Rouane-Hacene, Architecte-analyste Sénior pour m'avoir aidé dans l'élaboration des algorithmes, la conduite des expérimentations ainsi que la résolution de toutes les difficultés techniques relatives au développement de mon projet.

Je voudrais aussi exprimer ma profonde gratitude à ma femme Naima, ma fille Aya et mon fils Djebril pour lesquels je ne saurais jamais trouver les mots justes qui pourraient les réconforter pour leur patience exceptionnelle ou encore en retour de leur compréhension et amour infaillibles. Cependant, je leur demande pardon pour l'énorme sacrifice que je leur ai causé en me consacrant à l'aboutissement de ce projet.

Finalement, je remercie tous ceux qui m'ont soutenu, encouragé ou tout simplement cru en moi parmi mes amis et membres de ma famille, notamment mon frère Faycal.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

CHAPITRE 1

INTRODUCTION GÉNÉRALE

1.1 CONTEXTE

De nos jours, la construction et la maintenance des ontologies font l'objet d'une recherche très active vu le rôle central qu'occupent les ontologies dans la mise en œuvre des systèmes à base de connaissances et du Web sémantique. Une ontologie (Gruber, 1995; Guarino, 1998) est la spécification d'une conceptualisation d'un domaine. Autrement dit, c'est la formalisation des objets pertinents (dits instances), de leurs descripteurs (dits propriétés) ainsi que le regroupement en catégories (appelées concepts) de ces objets et leurs relations.

Le projet INUKHUK (Hacene et al., 2010) (initié au laboratoire GDAC de l'université UQAM) tente de mettre en œuvre une plateforme de génie ontologique facilitant les opérations de construction et de maintenance des ontologies de domaine telles que la construction par fusion de modules ontologique, extraction de modules, restructuration d'ontologies de faible qualité, évaluation de la qualité des ontologies, etc.

Notre travail de recherche intervient au sein d'un projet INUKHUK, plus particulièrement dans le cadre de la restructuration d'ontologies.

1.2 PROBLÉMATIQUE

La restructuration (Hacene et al., 2010) consiste à réorganiser la structure de l'ontologie de façon à augmenter sa qualité. Par exemple, optimiser la factorisation des propriétés (attributs ou relations) partagées entre plusieurs concepts ontologiques. La factorisation conduit souvent à la création de nouvelles abstractions ontologiques. Ces abstractions ne sont pas nommées et rendent l'ontologie moins compréhensible par un humain et donc non exploitable. Il faudra associer des noms de concepts intelligibles (Ang. labels) à ces nouvelles abstractions. Cette opération de fouille de noms pour les nouvelles abstractions est au centre de notre contribution au projet INUKHUK.

Il faut souligner que pour la restructuration des ontologies, la fouille de noms pour les nouveaux éléments ontologiques est une opération d'une extrême importance. En effet, la restructuration permet d'avoir une ontologie ayant une structure optimale (Par exemple, la réutilisation maximale des propriétés et des liens sémantiques) qui favorise son utilisation. Donner des noms aux nouveaux concepts qui sont cohérents avec leurs descriptions (attributs et relations) rend l'ontologie plus compréhensible par un humain qui sera capable donc de comprendre sa structure et de continuer à la maintenir. D'autre part, les méthodes d'alignement et d'évaluation de la qualité des ontologies (Miller, 1995; Suchanek et al., 2007) s'appuient généralement sur la comparaison de noms de concepts. De ce fait, une ontologie dont les concepts ont des noms appropriés et cohérents avec les descriptions associées favorise l'usage de ces méthodes et retourne de résultats plus précis.

1.3 OBJECTIFS

Les objectifs du projet s'inscrivent dans plusieurs volets :

— Volet théorique : consiste à faire une revue de la littérature sur le problème de nommage des nouvelles abstractions ontologiques et voir comment qu'il a été abordé dans la littérature et les solutions proposées. Cela nous aidera par la suite à concevoir une solution originale qui sera évaluée et intégrée à la plateforme INUKHUK.
— Volet algorithmique : consiste à traduire la solution du problème de nommage en algorithmes avec un focus sur les performances et la mise à l'échelle.
— Volet implémentation : consiste à développer un API de fouille de noms de concepts ainsi qu'une interface utilisateur (Ang. GUI) afin d'intégrer l'opération de nommage à la plateforme INUKHUK.

1.4 APPROCHE

Tout en restant ouvert à d'autres alternatives que nous avons découvert lors de notre analyse de l'état de l'art du problème de fouille de noms des concepts, nous partons d'une vision pour la solution du problème qui est fortement liée à l'approche de restructuration utilisée dans INUKHUK.

Cette approche, basée sur le cadre formelle de l'analyse Relationnelle de Concepts (ARC) (Hacene et al., 2013) qui à son tour s'appuie sur l'analyse Formelle de Concepts (AFC) (Ganter et Wille, 1999), prend en entrée un ensemble de concepts ontologiques avec leurs propriétés et produit en sortie de nouvelles abstractions de concepts qui reflètent tous les partages pertinents dans les propriétés. Chaque nouvelle abstraction de concepts possède deux descriptions :

— Description en extension : un ensemble des noms de concepts (En Anglais *Extent*).
— Description en intension : un ensemble de propriétés partagées par les concepts de la description en extension (En Anglais *Intent*).

Ainsi, pour chaque nouvelle abstraction de concepts, nous disposons de deux dimensions pouvant être utilisées de façon séparée ou combinée pour trouver un nom approprié à cette abstraction. À noter que les noms candidats issus de l'analyse des deux dimensions peuvent être confrontés à titre de validation.

L'analyse des deux ensembles intent et extent est réalisée de la façon suivante :

- Utilisation des ressources linguistiques et ontologiques (Wordnet, Wikipedia, Probase, etc.) pour la détermination d'un sujet général (Ang. general topic).
- Utilisation des algorithmes de calcul du plus petit terme commun entre un ensemble de termes (LCS)
- Utilisation de moteurs de recherche pour le calcul de fréquence de termes dans les documents en ligne
- Etc.

1.5 MÉTHODOLOGIE

Le travail est réalisé en plusieurs étapes. La durée moyenne d'une étape est de trois mois. Au bout de chaque étape un artéfact (document d'analyse, code java, etc.) est livré.

- **Étape 1 (Revue de la littérature et des outils rivaux) :** cette étape du projet consiste à spécifier la problématique de nommage, faire une étude laborieuse des approches de nommage existantes, les ressources linguistiques utilisées et les outils correspondants.
- **Étape 2 (Contribution théorique) :** mettre en place une approche de nommage adaptée au contexte de la restructuration des ontologies.
- **Étape 3 (Contribution algorithmique) :** écrire les algorithmes de nommage proposés à l'étape précédente.
- **Étape 4 (Développement et mise en œuvre) :** implémenter les algorithmes au sein de

l'outil TOPICMINER qui sera déployé avec la plateforme de restructuration INUKHUK.

— **Étape 5 (Expérimentation et validation) :** Exploiter les outils développés dans des projets de restructuration d'ontologies et évaluer les performances de ces outils.

1.6 PLAN DE MÉMOIRE

La table 1.1 illustre le plan de ce mémoire.

	Titre	Contenu
Chapitre 1	Introduction générale	On y trouve le contexte de la recherche effectuée ainsi que la problématique traitée.
Chapitre 2	Analyse formelle de concepts (AFC)	Ce chapitre introduit les fondements de l'AFC, les algorithmes de construction de treillis de concepts ainsi que les diverses applications de cette technique.
Chapitre 3	État de l'art	Présente une revue de littérature des différentes approches qui ont été utilisées pour résoudre la problématique proposée.
Chapitre 4	Annotation des concepts	Ce chapitre décrit notre approche pour la fouille de noms pour les concepts générés dans le cadre de la ré-ingénierie des ontologies au sein de la plateforme INUKHUK. On y trouve une description des fondements théoriques de notre approche ainsi que les algorithmes développés.
Chapitre 5	Expérimentation et validation	Ce chapitre décrit le protocole mis en place pour la validation de l'approche ainsi que les expérimentation réalisées.
Chapitre 6	Conclusion générale	Ce chapitre confronte les résultats obtenus aux objectifs établis tout au début du projet.

Tableau 1.1: Plan de mémoire.

CHAPITRE 2

ANALYSE FORMELLE DE CONCEPTS

Ce chapitre est une introduction au treillis de concepts d'une relation binaire (individus $\times$ propriétés) et à l'AFC. Nous présentons d'abord les notions fondamentales de cette théorie, puis nous abordons les différentes structures conceptuelles. Par la suite, nous décrivons le moyen de coder des données non binaires en AFC et le mécanisme qui s'y rattache. Nous allons aussi aborder les algorithmes de construction du treillis de concepts. À la fin du chapitre, nous décrivons quelques application usuelles.

2.1 FONDEMENTS DE L'AFC

Nous allons présenter les notions fondamentales de l'AFC, les structures conceptuelles qu'elle utilise et le mécanisme de scaling conceptuel qui permet de traiter les contextes non binaires comme nous allons le voir.

2.1.1 QUELQUES ÉLÉMENTS DE LA THÉORIE DES TREILLIS

Dans cette section, nous allons nous contenter d'exposer certaines notions fondamentales de la théorie des treillis [1].

2.1.1.1 Ensembles ordonnés

Dans ce qui suit, nous allons énoncer plusieurs définitions et propriétés qui vont servir à la proposition d'algorithmes pour la construction de treillis de concepts.

Définition 1 (ensemble ordonné) *Un couple $(E, \leq)$ formé d'un ensemble E et d'une relation d'ordre $\leq$ est appelé ensemble ordonné.*

Dans un ensemble ordonné $(E, \leq)$, pour une paire d'éléments $x, y \in E$, on dira que x et y sont comparables si $x \leq y$ ou $y \leq x$. Si tous les éléments sont comparables, on dit que l'ordre est total, sinon il n'est que partiel. Pour deux éléments comparables et différents, $x \leq y$ et $x \neq y$, on note $x < y$.

Les éléments d'un ensemble ordonné sont liés par la "*relation de couverture*" induite par l'ordre. Formellement, la couverture est définie de la manière suivante :

Définition 2 (relation de couverture) *Soient $(E, \leq)$ un ensemble ordonné et $x, y \in E$. On dit que y couvre x (ou y est un successeur de x, x est un prédécesseur de y), noté $x \prec y$, si $x < y, x \leq z < y \Rightarrow z = x$.*

Exemple : Dans l'ensemble $\mathbb{N}$ des entiers naturels, pour tout x, il n'existe aucun nombre entre x et $x+1$. Ainsi, le nombre $x+1$ couvre le nombre x.

1. Une excellente introduction aux ordres partiels et treillis peut être trouvée dans (Davey et Priestley, 1992) et dans (Ganter et Wille, 1999)

2.1.1.2 Diagramme de Hasse d'une relation d'ordre

Un aspect très utile des ensembles ordonnés est la représentation graphique. En effet, tout ensemble ordonné $(E,\leq)$ peut être représenté de façon schématique par un diagramme appelé "'diagramme de Hasse" qui est un graphe où les nœuds sont les éléments de E et les arcs représentent la relation de couverture entre éléments. La réalisation du diagramme consiste à :

- Étape (i) : $\forall x \in E$, associer un point $p(x)$ dans le plan cartésien,
- Étape (ii) : $\forall x, y \in E$, tel que $x \leq y$, tracer un segment $l(x,y)$ entre $p(x)$ et $p(y)$,
- Contraintes : faire l'étape (i) et (ii) en respectant les conditions suivantes :
  - Si $x \leq y$, alors l'ordonnée de $p(x)$ est inférieure à celle de $p(y)$,
  - Un point $p(z)$ ne peut se trouver sur le segment $l(x,y)$ si $z \neq x$ et $z \neq y$.

Exemple : Soit E l'ensemble formé par les diviseurs de 30. $E = \{1,2,3,5,6,10,15,30\}$ et $\leq$ la relation binaire "divise", notée $\div$. La Figure 2.1 montre le graphe associé à $\leq$.

	1	2	3	5	6	10	15	30
1	x	x	x	x	x	x	x	x
2		x			x	x		x
3			x		x		x	x
5				x		x	x	x
6					x			x
10						x		x
15							x	x
30								x

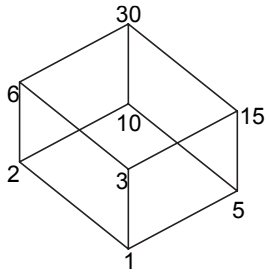


Figure 2.1: La relation $\div$ et le diagramme de Hasse de $(E,\div)$.

2.1.1.3 Éléments particuliers d'un ensemble ordonné

Définition 3 (Chaîne et antichaîne) *Une chaîne d'un ensemble ordonné $(E,\leq)$ est une partie S de E tel que quel que soit $(x,y) \in S^2, x \leq y$ ou $y \leq x$. Un ensemble tel que $x \leq y \Rightarrow x = y$ est appelé une antichaîne. Une chaîne S (resp. antichaîne) est dite maximale si et seulement si quel que soit l'élément $x \in E \setminus S$, l'ensemble $S \cup \{x\}$ n'est pas une chaîne (resp. antichaîne).*

Exemple : Sur la Figure 2.1, à droite, la séquence de nœuds $[1,5,15]$ forme une chaîne alors que la séquence $[6,10,15]$ forme une antichaîne.

Définition 4 (majorant - minorant) *Soient $(E,\leq)$ un ensemble ordonné et S une partie non vide de E. On dit qu'un élément a est un majorant (respectivement un minorant) de S, si pour tout x de S, $x \leq a$ (respectivement $x \geq a$).*

L'ensemble *S* peut avoir plusieurs *majorants* (respectivement *minorants*). Ainsi, nous appellerons *infimum* (respectivement *supremum*) les éléments dont la description correspond à la définition suivante :

Définition 5 (infimum - supremum) *Soient $(E,\leq)$ un ensemble ordonné et S une partie non vide de E. L'infimum, ou encore la borne inférieure, noté $\bigwedge$ [2] (respectivement supremum ou borne supérieure noté $\bigvee$ [3]) de S est le plus grand (respectivement le plus petit) élément, s'il existe, de l'ensemble des minorants (resp. des majorants) de S.*

Exemple : Dans la Figure 2.1, l'infimum $\bigwedge$ et le supremum $\bigvee$ représentent le plus grand commun diviseur (PGCD) et le plus petit commun multiple (PPCM), respectivement.

2. En anglais "meet"
3. En anglais "join"

Finalement, nous arrivons à la définition de la structure de treillis construit à partir d'un ensemble d'éléments et d'une relation d'ordre partiel.

Définition 6 (treillis) *Soit $(E,\leq)$ un ensemble ordonné. On dit que $(E,\leq)$ est un treillis si pour tout $x,\ y \in E$: $x \bigwedge y$ et $x \bigvee y$ existent.*

Définition 7 (treillis complet) *Le treillis E est complet si pour toute partie S non vide de E, $\bigvee S$ et $\bigwedge S$ existent.*

Exemple : Le treillis de la Figure 2.1 est un treillis complet.

Propriété 1 *Si E est un treillis fini alors il est complet.*

Définition 8 (demi-treillis) *Un ensemble ordonné $(E,\leq)$ est un sup-demi-treillis*[4] *(respectivement un inf-demi-treillis*[5]*) si tout couple d'éléments $x,y \in E$ admet un supremum $x \vee y$ (respectivement un infimum $x \wedge y$).*

2.1.1.4 Opérateurs de fermeture et correspondance de Galois

Un opérateur de fermeture sur un ensemble ordonné E est une application $\varphi : E \to E$, monotone croissante, extensive et idempotente. Formellement, elle est définie de la manière suivante :

Définition 9 (Opérateur de fermeture) *Soit $(E,\leq)$ un ensemble ordonné. On appelle opérateur de fermeture sur l'ensemble E toute application $\varphi : E \to E$ qui vérifie les trois propriétés suivantes :*

4. En anglais "join-semi-lattice" ou "sup-semi-lattice".
5. En anglais "meet-semi-lattice" ou "sub-semi-lattice".

— *idempotence :* $\forall x \in E,\ \varphi(\varphi(x)) = \varphi(x)$,

— *monotonie :* $\forall x, y \in E,\ x \leq y \Rightarrow \varphi(x) \leq \varphi(y)$,

— *extensivité :* $\forall x \in E,\ x \leq \varphi(x)$.

Définition 10 (fermé) *Étant donné un opérateur de fermeture* φ *sur un ensemble ordonné* $(E, \leq)$*, un élément* $x \in E$ *est dit "fermé" si* $x = \varphi(x)$*. L'élément x est aussi appelé le point fixe de la fermeture.*

Définition 11 (correspondance de Galois) *Soient* $(E, \leq_E)$ *et* $(F, \leq_F)$ *deux ensembles ordonnés. La paire de fonctions* $f : E \to F$ *et* $g : F \to E$ *définit une correspondance de Galois si la condition suivante est satisfaite :*

$$\forall x \in E, \forall y \in F, f(x) \leq y \Leftrightarrow g(y) \leq x.$$

2.1.1.5 Principe de la dualité

Définition 12 (Relation inverse) *Soit* $\leq$ *une relation binaire d'un ensemble E vers un ensemble F. On appelle relation inverse de* $\leq$*, notée* $\geq$*, la relation binaire de F vers E définie par :* $\forall (x, y) \in E \times F,\ x \leq y \Leftrightarrow y \geq x$.

Si $(E, \leq)$ est partiellement ordonné, la relation inverse $\geq$ est également un ordre partiel sur E. Le principe de la dualité permet de dériver les propriétés duales de $(E, \geq)$ à partir des propriétés de $(E, \leq)$.

Propriété 2 *Soit* $\geq$ *la relation inverse de* $\leq$*. Toute assertion sur* $\leq$, $\vee$, $\wedge$ *reste vraie en remplaçant* $\leq$ *par* $\geq$ *et en permutant* $\vee$ *et* $\wedge$.

2.1.2 NOTIONS FONDAMENTALES DE L'AFC

L'AFC (Ganter et Wille, 1999) étudie les structures partiellement ordonnées, connues sous le nom de *treillis de Galois* (Barbut et Monjardet, 1970) ou *treillis de concepts*, induites par une relation binaire sur une paire d'ensembles, appelés respectivement individus (*objets*) et propriétés (*attributs*). L'AFC permet d'identifier des groupements d'objets ayant des attributs en commun, appelés *concepts*, et de les organiser en une hiérarchie conceptuelle (*treillis de concepts*). Elle s'appuie sur les résultats de la théorie des treillis (voir section 2.1.1) dont les fondements mathématiques ont été posés par Birkhoff (Birkhoff, 1940).

2.1.2.1 Contextes Formels

Les données en AFC sont décrites par le biais d'un tableau booléen appelé "*contexte formel*". Formellement, un contexte formel est définit de la manière suivante :

Définition 13 (contexte formel mono-valué) *Un contexte formel est un triplet* $\mathscr{K} = (O, A, I)$ *où O est l'ensemble des objets formels, A est l'ensemble des attributs formels et I est une relation binaire, i.e.,* $I \subseteq O \times A$*, précisant pour chaque objet formel les attributs qui lui sont associés.*

Exemple : La table illustrée à gauche de la Figure 2.2 présente un contexte mono-valué, appelé "`Humain`", où les objets formels représentent des personnes et les attributs formels représentent certains traits personnels tels que l'âge et l'expérience de travail (e.t.).

	(a) age≤27	(b) age>27	(c) e.t.≤5	(d) e.t.>5
1	X		X	
2		X	X	
3	X		X	
4	X			X
5		X	X	
6		X		X

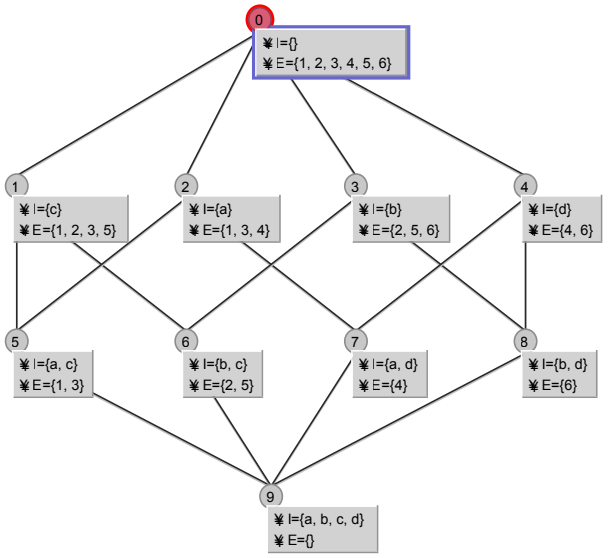


Figure 2.2: Gauche : Un contexte binaire . Droite : Le treillis de concepts associé.

2.1.2.2 Construction d'un contexte formel

Le format le plus simple pour décrire un contexte formel est un tableau binaire à deux dimensions où chaque ligne représente un objet formel et chaque colonne représente un attribut formel. L'intersection de la ligne o avec la colonne a contient une croix si et seulement si $(o,a) \in I$. Le type de données qui convient le plus pour la représentation en mémoire d'un contexte mono-valué est une matrice de bits [6].

Il n'y a pas de restrictions sur la nature des objets et des attributs. On peut considérer des objets physiques, des personnes, des nombres, des processus, des structures, etc. Il est possible de permuter le rôle des objets et des attributs : si $\mathscr{K} = (O,A,I)$ est un contexte, alors (O,A,I^{-1}) est son contexte dual avec $(o,a) \in I \Leftrightarrow (a,o) \in I^{-1}$.

6. Il n'y a pas de type de données standard pour la mémorisation d'un contexte car cela dépend des opérations que l'on veut réaliser qui consistent dans la plupart des cas à calculer les fermetures des ensembles d'objets, respectivement ceux des attributs

2.1.2.3 Correspondance de Galois dans un contexte formel

Étant donné un contexte formel $\mathscr{K} = (O,A,I)$, on définit les deux fonctions f et g sur ce contexte de la manière suivante :

Définition 14 *La fonction f associe à un ensemble d'objets X, l'ensemble d'attributs Y partagés par tous les objets de X ; tandis que g est la fonction duale pour les ensembles d'attributs :*

— $f : 2^O \to 2^A,\ f(X) = X' = \{a \in A \mid \forall o \in X, oIa\}$

— $g : 2^A \to 2^O,\ g(Y) = Y' = \{o \in O \mid \forall a \in Y, oIa\}$

La paire de fonctions (f,g), résumant les liaisons entre les objets et les attributs dans le contexte, définit une correspondance de Galois entre les deux ensembles ordonnés $(2^O,\subseteq)$ et $(2^A,\subseteq)$ (voir la définition 11 de la correspondance de Galois).

Propriété 3 *La paire de fonctions (f,g) de la définition 14 définie une correspondance de Galois entre 2^O et 2^A du contexte formel $\mathscr{K} = (O,A,I)$.*

Exemple : Dans le contexte à gauche de la Figure 2.2, $\{1,3\}' = \{a,c\}$ et $\{b,c\}' = \{2,5\}$.

2.1.2.4 Opérateurs de fermeture dans un contexte formel

Les opérateurs de composition $g \circ f$ et $f \circ g$ sont des opérateurs de fermeture sur les deux ensembles 2^O et 2^A. Par conséquent, chacun des deux opérateurs induit une famille d'ensembles *fermés*, notée $\mathscr{C}^o$ et $\mathscr{C}^a$ respectivement, avec f et g deux applications bijectives entre ces deux familles.

Propriété 4 *Soit un contexte formel $\mathscr{K} = (O,A,I)$. Les compositions des fonctions f et g : $f \circ g : (2^A, \subseteq) \to (2^A, \subseteq)$ et $g \circ f : (2^O, \subseteq) \to (2^O, \subseteq)$ sont les opérateurs de fermeture sur 2^A et 2^O, respectivement. Les deux opérateurs de composition sont notés par $''$.*

Ainsi, étant donné le contexte formel $\mathscr{K} = (O,A,I)$ et l'opérateur de fermeture $''$, et selon la définition 9, un sous ensemble d'objets $X \subseteq 2^O$ (respectivement un sous ensemble d'attributs $Y \subseteq 2^A$) est fermé si $X'' = X$ (respectivement Y"=Y).

Exemple : Dans le contexte de la Figure 2.2, les ensembles $\{1,3,4\}$ et $\{b,d\}$ sont fermés.

La famille des fermés des objets $\mathscr{C}^o$ et celle d'attributs $\mathscr{C}^a$ munies de la relation d'ordre partiel $\subseteq$ (inclusion ensembliste) constituent deux treillis complets, notés $\mathscr{L}^o$ et $\mathscr{L}^a$. De plus, la fonction $'$ (f ou g) est une bijection entre les deux familles $\mathscr{C}^o$ et $\mathscr{C}^a$ et définit un isomorphisme entre les treillis respectifs $\mathscr{L}^o$ et $\mathscr{L}^a$. À chaque fermé X de $\mathscr{C}^o$ correspond un fermé unique Y de $\mathscr{C}^a$, tel que $X' = Y$. Dans la section suivante nous allons caractériser ces paires d'éléments (X,Y) et décrire les différentes structures qui les englobent.

2.1.2.5 Concepts formels, extension et intension

Le couple (X,Y) d'ensembles fermés où X représente un sous-ensemble d'objets et Y un sous-ensemble d'attributs est appelé *concept (formel)*.

Définition 15 *Un concept formel d'un contexte $\mathscr{K}$ est une paire (X,Y) où $X \in 2^O$, $Y \in 2^A$, $X = Y'$ et $Y = X'$. L'ensemble X est appelé extension et Y l'intension du concept formel.*

Exemple : Dans le contexte de la Figure 2.2, le couple $(\{1,3,4\},\{a\})$ est un concept, alors que $(\{2,5\},\{b\})$ n'en est pas un.

Selon cette définition, un concept formel a deux dimensions : l'extension X et l'intension Y. Cette description est "redondante" car chaque dimension détermine l'autre (du fait que $Y = X'$ et $X = Y'$). Toutefois cette définition le rend conforme à la définition traditionnelle de concept en philosophie où "on nomme concept une idée ou représentation de l'esprit qui abrège et résume une multiplicité d'objets empiriques ou mentaux par abstraction et généralisation de traits communs identifiables". À noter que les deux dimensions suivent la loi de proportionnalité inverse : plus l'extension augmente, plus l'intension diminue et inversement.

D'un point de vue schématique, lorsqu'un contexte est décrit par une table binaire, chaque concept formel (X,Y) correspond à une sous table rectangulaire avec un ensemble de lignes X et un ensemble de colonnes Y non nécessairement contigües.

Les concepts formels d'un contexte correspondent aux rectangles maximaux de la table binaire associée. Pour visualiser ces rectangles, il est, parfois, nécessaire de procéder à des permutations de lignes et/ou de colonnes. Ces permutations n'affectent pas le contexte du moment où il n'y a pas d'ordre entre les éléments des deux ensembles O et A. Un contexte a en général une multitude de concepts formels. Leur nombre peut être exponentiel en la taille du contexte. L'ensemble de tous les concepts formels d'un contexte $\mathscr{K} = (O,A,I)$ sera noté ici par $\mathscr{C}_{\mathscr{K}}$. Plusieurs algorithmes ont été proposés pour le calcul de cet ensemble ou encore la relation d'ordre entre les éléments de cet ensemble. Par la suite, nous allons présenter deux approches pour le calcul de l'ensemble $\mathscr{C}_{\mathscr{K}}$ d'un contexte $\mathscr{K}$ donné.

2.1.2.6 Hiérarchies conceptuelles

La famille $\mathscr{C}_{\mathscr{K}}$ des concepts formels d'un contexte $\mathscr{K} = (O,A,I)$ est partiellement ordonnée par la relation d'inclusion $\subseteq$ entre intensions/extensions.

Définition 16 *Étant donné* (X_1,Y_1) *et* (X_2,Y_2) *deux concepts formels d'un contexte* $\mathscr{K} = (O,A,I)$. *Le concept* (X_1,Y_1) *est un* ***sous-concept*** *de* (X_2,Y_2) *(d'une manière équivalente,* (X_2,Y_2) *est un* ***super-concept*** *de* (X_1,Y_1)*) si et seulement si* $X_1 \subseteq X_2$ *(*$Y_2 \subseteq Y_1$*). On utilise le signe* $\leq$ *pour exprimer cette relation. On obtient :* $(X_1,Y_1) \leq_{\mathscr{K}} (X_2,Y_2) \Leftrightarrow X_1 \subseteq X_2$*(ou bien* $Y_2 \subseteq Y_1$*).*

Un super-concept direct (respectivement le sous-concept direct) d'un concept est aussi appelé "*successeur*" direct (respectivement prédécesseur direct).

Définition 17 (successeur direct) *Soit* $c = (X,Y)$ *un concept et* $\bar{c} = (\bar{X},\bar{Y})$ *un de ses super-concepts.* $\bar{c}$ *est dit successeur direct de c si et seulement si il n'existe pas un autre concept* $\hat{c} = (\hat{X},\hat{Y})$ *tel que* $Y \subseteq \hat{Y} \subseteq \bar{Y}$.

On définit la face d'un concept par rapport à un successeur direct, notée $\delta(c)$, de la manière suivante :

Définition 18 (face d'un concept) *Soient* $c = (X,Y)$ *un concept et* $\bar{c} = (\bar{X},\bar{Y})$ *son successeur direct.* $\delta(c) = Y - \bar{Y}$.

La famille des concepts organisée par la relation de super-concept (sous-concept) forme un treillis de concepts.

Propriété 5 (treillis de concepts) *L'ensemble de tous les concepts formels de* $\mathscr{K} = (O,A,I)$*, ordonné par la relation de super-concept (sous-concept) est un treillis complet. Il est noté* $\mathscr{L} = \langle \mathscr{C}_{\mathscr{K}}, \leq_{\mathscr{K}} \rangle$ *et est appelé le treillis de concepts du contexte* $\mathscr{K}$.

Exemple : Le treillis de concepts construit à partir du contexte de la Figure 2.2 est présenté sur la même figure, à droite. À noter que les extensions et les intensions sont montrées sur les nœuds représentant les concepts et identifiables par les ensembles “E” et “I”, respectivement.

Le treillis de concepts est muni des opérations dites *infimum* et *supremum* qui sont notées, respectivement, par $\bigwedge$ et $\bigvee$.

Théorème 1 *L'ordre partiel $\mathscr{L} = \langle \mathscr{C}_{\mathscr{K}}, \leq_{\mathscr{K}} \rangle$ forme un treillis complet où l'infimum et le supremum sont définis comme suit :*

— $\bigwedge_{i=1}^{k}(X_i, Y_i) = (\bigcap_{i=1}^{k} X_i, (\bigcup_{i=1}^{k} Y_i)'')$,

— $\bigvee_{i=1}^{k}(X_i, Y_i) = ((\bigcup_{i=1}^{k} X_i)'', \bigcap_{i=1}^{k} Y_i)$.

Exemple : Dans le treillis de la Figure 2.2, l'infimum des deux concepts $(\{1,2,3,5\},\{c\})$ et $(\{1,3,4\},\{a\})$ est le concept $(\{1,3\},\{a,c\})$ tandis que le supremum des deux concepts $(\{2,5\},\{b,c\})$ et $(\{6\},\{b,d\})$ est le concept $(\{2,5,6\},\{b\})$.

La couverture supérieure (respectivement inférieure) d'un concept appartenant au treillis de concepts est définie comme suit :

Définition 19 (couverture inférieure - supérieure) *La couverture supérieure (respectivement inférieure) d'un concept est l'ensemble de tous ses super-concepts (respectivement sous-concepts).*

Une manière de présenter le treillis de concepts qui s'avère très utile en génie logiciel et dans la visulation des treillis de grande taille consiste à supprimer les redondances d'attributs et d'objets dans les intensions et les extensions des concepts, respectivement. En effet, pour un concept $c = (X,Y)$, X est présent dans tous les ancêtres de c et symétriquement, Y apparaît dans

tous ses descendants. Ainsi, la suppression des redondances consiste à retirer de l'extension d'un concept tous les objets formels (respectivement attributs formels) qui apparaîssent dans les extensions (respectivement intensions) des concepts de sa couverture supérieure (respectivement inférieure). La structure obtenue est appelée *treillis d'héritage* ou encore le *treillis des sommets simplifiés*. La Figure 2.3 montre le treillis d'héritage qui correspond au treillis de la Figure 2.2.

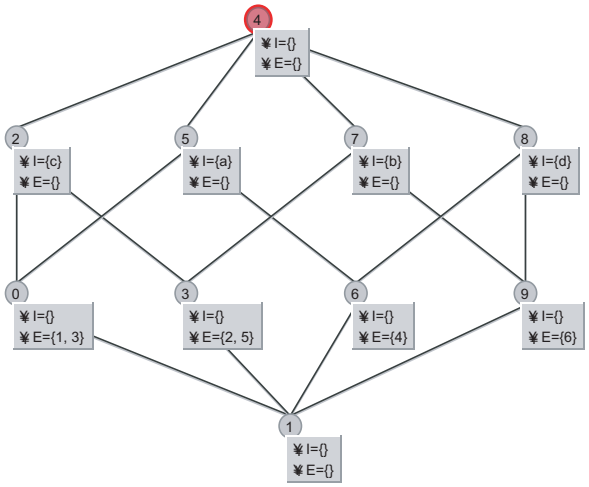


Figure 2.3: Le treillis d'héritage correspondant au treillis de la Figure 2.2.

Concept objet/attribut Dans le treillis de la Figure 2.2, on remarque que chaque objet o est attaché à un unique concept, appelé *concept objet* qui représente le concept minimal (en terme de nombre d'objets) dans le treillis ayant l'objet o dans son extension. De manière similaire, à chaque attribut a correspond un *concept attribut*, noté $\mu(a)$, qui représente le concept maximal dans le treillis ayant l'attribut a dans son intension.

Étant donné un contexte binaire, la paire de fonctions (γ,μ) permet d'obtenir ces concepts particuliers.

Définition 20 *Soit $\mathcal{K} = (O,A,I)$ un contexte binaire et $\mathcal{L} = \langle \mathcal{C}_{\mathcal{K}}, \leq_{\mathcal{K}} \rangle$ le treillis complet*

associé.

$$\gamma : O \to \mathscr{C}_{\mathscr{K}},\ \gamma(o) = (\{o\}'', \{o\}')$$

Définition 21 *Soit $\mathscr{K} = (O, A, I)$ un contexte binaire et $\mathscr{L} = \langle \mathscr{C}_{\mathscr{K}}, \leq_{\mathscr{K}} \rangle$ le treillis complet associé.*

$$\mu : A \to \mathscr{C}_{\mathscr{K}},\ \mu(a) = (\{a\}', \{a\}'').$$

Exemple : Dans le treillis de la Figure 2.2, $\gamma(\{2\}) = (\{2,5\}, \{b,c\})$, et $\mu(\{d\}) = (\{4,6\}, \{d\})$.

2.1.3 DONNÉES NON BINAIRES

2.1.3.1 Ensemble étendu d'attributs

Depuis l'adoption des treillis de concepts comme un outil d'analyse de données, leur cadre théorique a fait l'objet de nombreuses extensions dont l'intérêt consiste à remplacer les descriptions binaires des individus (conjonction d'attributs binaires) par d'autres formes de descriptions plus expressives. Le but est de pouvoir traiter des données concrêtes telles que celles de type objet-attribut-valeurs issues des bases de données et de la modélisation des problèmes concrets.

Ainsi, beaucoup de travaux (Diday et Emillion, 1997; Girard, 1997; Kent, 1996) ont émergé explorant les possibilités de traduire les différents types de données en variables binaires, notamment ceux basés sur les *échelles conceptuelles* (Ganter et Wille, 1999). D'autres travaux ont reconsidéré la définition même de certaines constructions fondamentales de la théorie des treillis, telles que la *correspondance de Galois*, dans le cas où les objets admettent des

structures complexes, éventuellement floues (Girard, 1997), probabilistes (Diday et Emillion, 1997) ou ensembles bruts (Kent, 1996). En AFC, les attributs non binaires (e.g., numériques, ordinaux, catégoriques, etc.) sont formulés à travers des contextes pluri-valués (Ganter et Wille, 1999).

Définition 22 (contexte pluri-valué) *Un contexte pluri-valué est un quadruplet* $\mathscr{K} = (O, A, V, I)$ *où* O *et* A *sont des ensembles d'objets et d'attributs respectivement,* V *est un ensemble de valeurs, et* $I \subseteq O \times A \times V$ *est une relation ternaire ayant des tuples sous la forme* (o, a, v) *qui signifie "l'objet o a la valeur v pour l'attribut a".*

La Table 2.1 montre un contexte pluri-valué, appelé *Humain*. Il est composé de six objets (1, 2, …, 6) représentant des individus et deux propriétés, *âge* et *travail* exprimant, respectivement, l'âge et le nombre d'années d'expérience de chaque individu.

	o_1	o_2	o_3	o_4	o_5	o_6
âge	25	28	22	26	33	35
travail	4	2	1	8	4	10

Tableau 2.1: Contexte pluri-valué *Humain*.

Dans le but de traiter un contexte pluri-valué $\mathscr{K} = (O, A, V, I)$, celui-ci est d'abord transformé en un contexte mono-valué (binaire) équivalent $\mathscr{K}^d$ appelé aussi le contexte *dérivé*. Pour ce faire, l'AFC possède une technique, appelée "*scaling*", que nous allons décrire dans la section suivante.

2.1.3.2 Scaling conceptuel

D'une manière générale, il n'y a pas un moyen "automatique" pour dériver des structures conceptuelles à partir de contextes pluri-valués. La première approche qui consiste à dériver

des treillis à partir de ce genre de contextes a été proposée par Ganter *et al.* dans (Ganter et al., 1986). Dès lors, une grande variété d'applications s'est appuyée sur cette méthode notamment celles qui traitent des données brutes en provenance des bases de données (Prediger, 1997).

Le scaling conceptuel (Ganter et Wille, 1999) fournit un cadre théorique complet pour la transformation de n'importe quel contexte pluri-valué en un contexte binaire équivalent. En effet, le processus de scaling associe à chaque attribut pluri-valué un contexte binaire appelé *échelle conceptuelle*.

Échelle

Définition 23 (Échelle) *Une **échelle** pour un attribut a d'un contexte pluri-valué donné est un contexte mono-valué $\mathscr{K}_a = (V_a, P_a, I_a)$ où les objets formels V_a sont les valeurs de l'attribut a tandis que les attributs formels P_a sont des propriétés distinguées de ces valeurs. Les objets de l'échelle sont appelés **valeurs de l'échelle** alors que les attributs sont appelés **attributs de l'échelle**.*

Quelques contextes élémentaires sont souvent utilisés comme des échelles. Nous citons à titre illustratif l'échelle nominale, ordinale, biordinale et dichotomique (Ganter et Wille, 1999). Par exemple, l'échelle nominale, notée $\mathscr{N}_n = (n, n, =)$, est utilisée avec les attributs pluri-valués dont les valeurs s'excluent mutuellement. Un attribut pluri-valué pouvant avoir les valeurs masculin, féminin ou neutre doit subir un scaling nominal. Ainsi les extensions des concepts définissent une partition de l'ensemble d'objets de l'échelle tel que illustré à droite de la Figure 2.2.

Contextes dérivés par des échelles conceptuelles À chaque échelle correspondant à un attribut a est associé un contexte $\mathscr{K}_a = (V_a, P_a, J_a)$ où les valeurs v_i de l'attribut a sont les

	1	2	3	4
1	X			
2		X		
3			X	
4				X

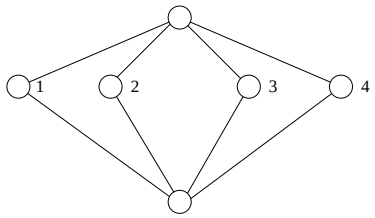


Tableau 2.2: Gauche : Échelle nominale $\mathcal{N}_4$. Droite : Le treillis associé (Adapté de (Ganter et Wille, 1999)).

objets formels tandis que les propriétés essentielles de ces valeurs deviennent les attributs formels a_j^d. À titre d'exemple, dans les échelles ordinales, les propriétés sont des prédicats permettant de comparer les valeurs à une constante dans un domaine (exemple, "age ≥ 20").

Les échelles correspondent aux différentes dimensions de la description de l'objet formel. Essentiellement, une échelle décrit d'une manière hiérarchique des abstractions utiles d'une dimension donnée, appelées *concepts de l'échelle*. Ces abstractions correspondent à des sous-ensembles de valeurs du co-domaine de l'attribut a (par exemple,"âge ≥ 20 et ≤ 27"), notés $cod(a) = V_a$. Par la suite, nous allons noter l'extension de l'attribut de l'échelle a_j^d, en terme de valeurs, par $a_j^{d'}$.

Treillis de l'échelle Le treillis de l'échelle $\mathscr{L}_a$ qui correspond au contexte $\mathscr{K}_a$ permet un regroupement des valeurs de l'attribut en sous-ensembles significatifs pour l'utilisateur. Les concepts échelle dans $\mathscr{L}_a$ représentent les sous-ensembles de valeurs de a. Dans le cas où la cardinalité de l'ensemble $cod(a)$ est élevée (cas des valeurs d'une colonne dans une base de données), le treillis $\mathscr{L}_a$ peut être difficile à obtenir d'une manière directe, c'est à dire, à partir de $\mathscr{K}_a$. Toutefois, la relation d'inclusion entre les extensions des différents attributs de l'échelle peut être utilisée pour restaurer la structure de $\mathscr{L}_a$. Dans ce cas particulier, le contexte formel utilisé est $\mathscr{K}_a^f = (P_a, P_a, \leq_f)$, où $a_j^d \leq_f a_l^d$ si et seulement si $a_j^{d'} \subseteq_f a_l^{d'}$.

Exemple : La Figure 2.4 présente des treillis échelles pour les attributs pluri-valués *âge* et *experience de travail* correspondant au contexte *Humain*. Les seuils 27 et 5 ont été arbitrairement choisis pour les attributs *âge* et *travail* respectivement.

Figure 2.4: Échelles pour *âge* et *expérience de travail* suivant la Table 2.1.

Scaling Le but de tout processus de scaling en AFC est l'obtention, à partir d'un contexte pluri-valué $\mathscr{K} = (O, A, V, I)$, le contexte formel dérivé $\mathscr{K}^d$ ayant les mêmes objets que le contexte de départ et des attributs en provenance de différentes échelles représentant des propriétés "significatives". Le mot "significatives" fait référence à l'interprétation des données qui ne peut être établie que par un expert du domaine. Cette interprétation est toujours faite dans un but et fondée sur des considérations théoriques (Prediger, 1997).

Le principe du scaling conceptuel consiste à affecter à chaque attribut pluri-valué $a \in A$ une échelle conceptuelle $\mathscr{K}_a$ qui est un contexte formel mono-valué $\mathscr{K}_a = (V_a, P_a, I_a)$. Ainsi, pour chaque objet, la valeur de l'attribut non binaire a est remplacée par un ensemble de valeurs binaires, a_i^d. Un objet o de $\mathscr{K}$ obtient a_i^d chaque fois que a_i^d inclut la valeur de a dans o. Le résultat est un contexte dérivé $\mathscr{K}^d$ qui peut être considéré comme étant une concaténation (une apposition comme dans (Ganter et Wille, 1999)) d'un ensemble de contextes $\mathscr{K}_a^d$, un par attribut pluri-valué. À partir du contexte dérivé, on peut construire le treillis de concepts de manière classique comme en (Wille, 1982). Le treillis du contexte dérivé est considéré comme étant la structure conceptuelle du contexte pluri-valué.

Exemple : Le contexte à gauche de la Figure 2.2, appelé $Humain^d$, a été obtenu par le scaling

du contexte *Humain* de la Table 2.1 en utilisant les échelles conceptuelles présentées dans la Figure 2.4.

Construction du treillis par scaling Le treillis $\mathscr{L}^d$ associé au contexte dérivé $\mathscr{K}^d$ peut être obtenu directement à partir de ce contexte en utilisant n'importe quelle procédure usuelle de construction de treillis. Il est évident que la forme exacte du treillis $\mathscr{L}^d$ dépend des échelles choisies pour les différents attributs muti-valués. Par exemple, le treillis du contexte dérivé $Human^d$ est donné à droite de la Figure 2.2.

Un autre intérêt des échelles conceptuelles réside dans la possibilité de les utiliser pour la visualisation des treillis, en particulier ceux de grande taille (comme dans le système TOSCANA (Vogt et Wille, 1994)). De plus, le treillis $\mathscr{L}^d$ peut être obtenu directement à partir des treillis $\mathscr{L}_a^d$ associés aux différents attributs pluri-valués du contexte initial $\mathscr{K}$ en utilisant une version légèrement modifiée de l'algorithme ASSEMBLER décrit dans (Valtchev et Missaoui, 2000). Par exemple, dans le cas du contexte dérivé $Humain^d$, les correspondances entre les treillis échelles de ses attributs pluri-valués *age* et $travail$ (voir Figure 2.4) et le treillis $\mathscr{L}^d$ qui lui correspond (voir Figure 2.2, à droite) sont faciles à établir.

2.2 CONSTRUCTION DU TREILLIS DE GALOIS

La construction du treillis à partir du contexte a toujours été un défi calculatoire vu que le nombre de concepts dans le treillis peut être exponentiel en fonction du nombre d'attributs ou d'objets du contexte. Toutefois, en pratique, seulement un petit nombre de concepts sont produits d'où l'intérêt de chercher des méthodes permettant de découvrir d'une manière efficace les concepts et, si nécessaire, de les organiser en une hiérarchie.

La construction d'un treillis est composée de deux tâches principales qui peuvent être exécutées

de manière simultanée ou séquentielle, à savoir, la découverte des concepts et le calcul de la relation de couverture. Dans la littérature de l'AFC, il y a une grande variété d'algorithmes [7] dédiés à ces deux tâches séparément ou conjointement. Toutefois, une distinction majeure entre ces algorithmes réside dans la manière d'acquérir les données d'entrée. Suivant la dichotomie actuelle, trois paradigmes d'algorithmes sont considérées :

— **Algorithmes batch** (Bordat, 1986; Nourine et Raynaud, 1999; Ganter, 1984) : représente la première génération des algorithmes de construction de treillis. Ils considèrent que les données (contexte formel) sont connues à l'avance. L'évolution des données (ajout objet/attribut au contexte) entraîne la reconstruction du treillis à nouveau.

— **Algorithmes incrémentaux** (Godin et al., 1995; Valtchev et Missaoui, 2000) : sont apparus pour remédier au problème de la reconstruction du treillis dans le cadre de contextes dynamiques. En effet, suite à une modification du contexte, ces algorithmes effectuent des mises à jour locales du treillis correspondant. À noter qu'un algorithme incrémental peut simuler un algorithme batch en réalisant une suite d'ajouts d'objets/attributs à un contexte initialement vide.

— **Algorithmes d'assemblage** (Valtchev et Missaoui, 2000) : Les travaux (Godin et al., 1995) sur les performances des algorithmes incrémentaux ont abouti à une troisième catégorie d'algorithmes qui généralise le caractère incrémental à des ensembles d'objets/attributs. Ces algorithmes se basent sur l'apposition/sous-position de contextes et les demi-produits de treillis, pour définir une opération binaire sur les treillis complets, appelée ASSEMBLAGE permettant de construire un treillis à partir des deux treillis associés aux deux parties du contexte. Cette technique peut être utilisée dans le cadre de la stratégie "*diviser pour régner*" de construction de treillis.

Les sous structures de treillis, mentionnées plus haut, ont aussi bénéficié de développements

7. Voir (Kuznetsov et Ob'edkov, 2000) pour une étude comparative.

algorithmiques. Ainsi, des techniques particulières ont été élaborées pour ce type de structures permettant de traiter des problèmes spécifiques à un faible coût. Concernant les SHG par exemple, un ensemble d'algorithmes efficaces, batch et incrémentaux, ont été proposés dans la littérature tels que ceux de Godin (Godin et Valtchev, 2003). Les icebergs quant à eux restent proches des treillis complets et par conséquent peu d'algorithmes ont été, explicitement, conçus pour eux. Parmi les méthodes dédiées aux icebergs citons TITANIC (Stumme et al., 2002) et la paire d'algorithmes appelée MAGALICE (MAGALICE-O (Rouane, 2003) et MAGALICE-A (Nehme et al., 2005).

Les algorithmes incrémentaux considèrent le tableau de la relation binaire une ligne à la fois. Cette approche est fort intéressante parce qu'on n'a pas besoin de recalculer de nouveau les éléments du treillis de Galois chaque fois qu'un nouvel objet est considéré. Par conséquent, les algorithmes incrémentaux permettent beaucoup plus le maintient de l'intégrité du treillis suite à l'ajout d'un nouvel objet/attribut au contexte que la construction proprement dite du treillis.

Godin *et al.* (Godin et al., 1995) ont proposé une procédure incrémentale qui permet de modifier de manière locale la structure d'un treillis (insérer de nouveaux concepts, compléter des concepts existants, détruire des liens redondants, etc.) tandis qu'une grande partie du treillis reste inchangée. Un autre algorithme incrémental a été conçu par Carpineto et Romano (Carpineto et Romano, 1996).

2.2.1 INCRÉMENTALITÉ PAR OBJET

Une manière de construire le treillis $\mathscr{L}$ consiste à ajouter progressivement un objet o_i au treillis $\mathscr{L}_{i-1}$ (qui correspond au contexte $\mathscr{K} = (\{o_1, ..., o_{i-1}\}, A, I)$), en commençant par le seul objet o_1 et en faisant à chaque étape (ajout) une série de mises à jour structurelles. Dans ce qui suit, nous allons considérer l'exemple de l'ajout du nouvel objet $(\{9\}, \{cdfgh\})$ au

contexte formel de la Table 2.3 dont le treillis est illustré par la Figure 2.5.

	a	b	c	d	e	f	g	h
1	0	1	1	1	0	0	1	1
2	1		1	0	0	0	0	0
3	0	0	0	1	1	1	1	1
4	0	0	0	0	0	0	1	0
5	0	0	0	0	1	1	0	1
6	1	1	1	1	0	0	0	0
7	0	1	1	1	0	0	0	0
8	0	0	0	1	0	0	0	0

Tableau 2.3: Un contexte formel.

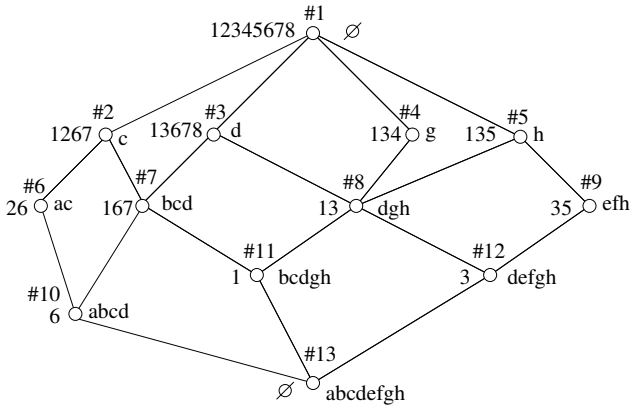


Figure 2.5: Le treillis de concepts du contexte de la Table 2.3.

2.2.1.1 Fondements théoriques :

Nous allons rappeler les aspects théoriques importants de l'approche incrémentale définie dans (Valtchev et al., 2002) et (Valtchev et al., 2003a) nécessaires à la mise au point des algorithmes incrémentaux de construction de treillis de concepts. Pour plus de détails, le lecteur est invité à consulter les deux sources principales sus-citées.

Soient le contexte $\mathscr{K} = (O, A, I)$ et le contexte $\mathscr{K}^+ = (O^+, A, I^+)$ obtenus par l'insertion du

nouvel objet o dans $\mathscr{K}$ tel que $O^+ = O \cup \{o\}$ et $I^+ = I \cup (\{o\} \times \{o\}')$. Le treillis $\mathscr{L}$=$\langle \mathscr{C}, \leq \rangle$ (respectivement $\mathscr{L}^+$=$\langle \mathscr{C}^+, \leq^+ \rangle$) dénote le treillis qui correspond au contexte $\mathscr{K}$ (respectivement $\mathscr{K}^+$).

L'approche incrémentale s'appuie sur le fait que si $c = (X, Y)$ est un concept de $\mathscr{L}$, alors $Y \cap \{o\}'$ est fermé et correspond à un concept de $\mathscr{L}^+$.

Propriété 6 *: L'ensemble $\mathscr{C}^a$ est fermé par l'intersection.*

La mise à jour d'un treillis $\mathscr{L}$ suite à l'ajout du nouvel objet o au contexte associé consiste à calculer l'intersection des intensions des concepts de $\mathscr{L}$ avec $\{o\}'$. De ce fait, deux types d'intersections surgissent : celles qui existent déjà dans $\mathscr{L}$ et celles qui sont nouvelles. Ainsi, les concepts de $\mathscr{L}$ sont répartis en trois catégories : les concepts dits "*modifiés*", notés $\mathbf{M}(o)$, qui sont le résultat des intersections existantes, les concepts dits "*inchangés*", notés $\mathbf{U}(o)$, qui demeurent inchangés, et les concepts dits "*géniteurs*", notés $\mathbf{G}(o)$, qui donnent naissance à de nouveaux concepts, notés $\mathbf{N}^+(o)$ (nouvelles intersections). Dans $\mathscr{L}^+$ ces mêmes ensembles sont notés $\mathbf{M}^+(o)$, $\mathbf{U}^+(o)$ et $\mathbf{G}^+(o)$, respectivement.

Exemple : La Figure 2.6 montre le treillis $\mathscr{L}^+$ obtenu suite à l'ajout de l'objet $(\{9\}, \{cdfgh\})$ [8]. Les modifiés, géniteurs et inchangés dans $\mathscr{L}$ et $\mathscr{L}^+$ sont :

— $\mathbf{G}(9) = \{c_{\#7}, c_{\#9}, c_{\#11}, c_{\#12}, c_{\#13}\}$,

— $\mathbf{M}(9) = \{c_{\#2}, c_{\#3}, c_{\#4}, c_{\#5}, c_{\#8}\}$,

— $\mathbf{U}(9) = \{c_{\#10}, c_{\#6}\}$,

— $\mathbf{G}^+(9) = \{c_{\#7}, c_{\#9}, c_{\#11}, c_{\#12}, c_{\#13}\}$,

— $\mathbf{M}^+(9) = \{c_{\#2}, c_{\#3}, c_{\#4}, c_{\#5}, c_{\#8}\}$,

— $\mathbf{N}^+(9) = \{c_{\#14}, c_{\#15}, c_{\#16}, c_{\#17}, c_{\#18}\}$,

— $\mathbf{U}^+(9) = \mathbf{U}(9)$.

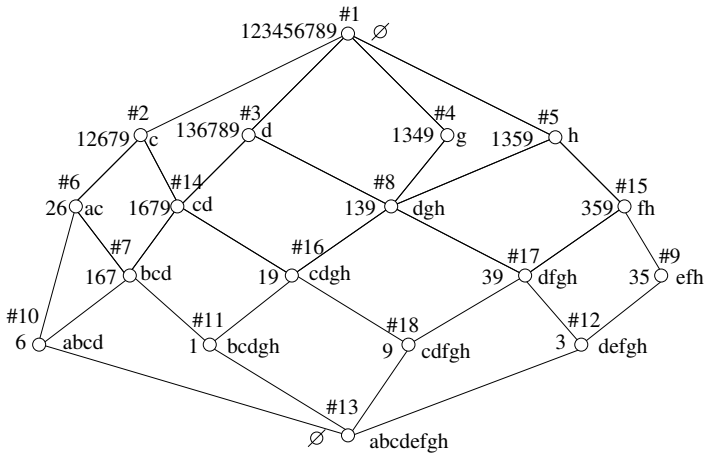


Figure 2.6: Le treillis $\mathscr{L}^+$ obtenu après restructuration du treillis $\mathscr{L}$ de la figure 2.5 suite à l'ajout du nouvel objet $(\{9\},\{cdfgh\})$.

La tâche de restructuration du treillis $\mathscr{L}$ consiste à déterminer les ensembles de géniteurs $\mathbf{G}(o)$ afin de pouvoir créer les nouveaux concepts et les modifiés $\mathbf{M}(o)$ afin de mettre à jour les extensions correspondantes.

2.2.1.2 Description de l'algorithme

La méthode (Algorithme 1) proposée dans (Valtchev et al., 2003a) permet de mettre à jour un treillis suite à l'ajout d'un nouvel objet. Les opérations de mise à jour consistent à : (i) identifier les classes d'équivalence, (ii) trouver l'élément maxima de chaque classe $[]_{\mathscr{Q}}$ et de déterminer son statut (géniteur ou modifié), (iii) mettre à jour des concepts modifiés, (iv) créer les nouveaux concepts, (v) déterminer les nouvelles couvertures inférieures et supérieures des nouveaux concepts, (vi) et finalement éliminer les liens obsolètes de chaque générateur. Ces opérations sont réalisées de la manière suivante :

— **Détection des maxima de classes d'équivalence $[]_{\mathscr{Q}}$:** Comme seulement l'élément

8. Par mesure de lisibilité, nous donnerons les ensembles d'objets/attributs sans séparateurs.

maxima est utilisé dans la restructuration du treillis initial, il est inutile de construire toute la classe d'équivalence. En effet, selon la propriété 7, pour tout concept c non maximal dans sa classe d'équivalence, il existe toujours un successeur $\bar{c}$ appartenant la classe $[c]_{\mathscr{Q}}$. Par conséquent, le statut d'un concept peut être déterminé en le comparant avec ses successeurs qui sont visités préalablement dans le cas d'un parcours descendant du treillis.

Propriété 7 (détection des maxima des classes) *Toutes les classes* $[]_{\mathscr{Q}}$ *dans* $\mathscr{L}$ *sont des ensembles convexes :* $\forall\, c,\, \bar{c},\, \underline{c} \in \mathscr{C},\, \underline{c} \leq c \leq \bar{c}$ *et* $[\bar{c}]_{\mathscr{Q}} = [\underline{c}]_{\mathscr{Q}} \Rightarrow [\bar{c}]_{\mathscr{Q}} = [c]_{\mathscr{Q}}$.

— **Calcul des couvertures supérieures des nouveaux concepts :** De plus, pour le calcul des couvertures supérieures d'un nouveau concept c, au lieu de considérer tous les successeurs potentiels, on peut prendre l'ensemble $\{\chi^+(\bar{c}) \mid \bar{c} \in Cov^u(\gamma(c))\}$, c'est à dire, les minimas des images par χ^+ de tous les successeurs du géniteur $\gamma(c)$.

Propriété 8 *Pour tout* $c = (X,Y) \in \mathscr{L}$, *et* $\bar{c}_1 = (\bar{X}_1,\bar{Y}_1), \bar{c}_2 = (\bar{X}_2,\bar{Y}_2) \in Cov^u(c)$, $\bar{X}_1 \cap \bar{X}_2 = X$.

— **Destruction des liens obsolètes :** Enfin tout modifié $\hat{c}$ qui est un successeur direct d'un générateur $\bar{c}$ dans $\mathscr{L}$ doit être déconnecté de $\bar{c}$ dans $\mathscr{L}^+$ car $\chi^+(hatc)$ est nécessairement un successeur du nouveau concept produit par $\chi^+(barc)$.

Propriété 9 *Pour tout* $\bar{c} \in \mathbf{G}(o)$, $\hat{c} \in \mathbf{M}(o)$ *:* $\bar{c} \prec \hat{c} \;\Rightarrow\; \hat{c} \in \min(\{\chi^+(\hat{c}) \mid \hat{c} \in Cov^u(\bar{c})\})$.

L'algorithme 1 prend un treillis $\mathscr{L}$ et un nouvel objet [9] et produit le treillis mis à jour $\mathscr{L}^+$. Les valeurs de $calQ$ et χ^+ sont représentées par une structure générique (la structure *ChiPlus*) indexée par les identificateurs de concepts. Au début, les concepts sont triés par ordre croissant de la taille de l'intension afin de permettre un parcours descendant du treillis (la routine SORT de la ligne 3). Le traitement itératif (lignes 4 jusqu'à ligne 20) examine chaque concept c

9. L'ensemble A est supposé être connu, c'est à dire, $\{o\}' \subseteq A$.

```
 1: procedure ADD-OBJECT(In/Out : 𝒟 = ⟨𝒞, ≤⟩ a lattice ; In : o an object)
 2:
 3: SORT(𝒞)
 4: for all c in 𝒞 do
 5:    new-max ← ARGMAX({|𝒬(c̄)| | c̄ ∈ Cov^μ(c)})
 6:    if |𝒬(c)| ≠ |𝒬(new-max)| then
 7:       if |𝒬(c)| = |Intent(c)| then
 8:          Extent(c) ← Extent(c) ∪ {o}     {c is modified}
 9:          M(o) ← M(o) ∪ {c}
10:          new-max ← c
11:       else
12:          ĉ ← NEW-CONCEPT(Extent(c) ∪ {o},𝒬(c))     {c is genitor}
13:          Candidates ← {ChiPlus(c̄) | c̄ ∈ Cov^μ(c)}
14:          for all c̄ in MIN-CLOSED(Candidates) do
15:             NEW-LINK(ĉ,c̄)
16:             if c̄ ∈ M(o) then
17:                DROP-LINK(c, c̄)
18:          new-max ← ĉ
19:          𝒟 ← 𝒟 ∪ {ĉ}
20:    ChiPlus(c) ← new-max
```

Algorithm 1: Mise à jour du treillis suite à l'ajout d'un nouvel objet au contexte.

du treillis $\mathscr{L}$ et détermine son statut dans $[c]_{\mathscr{Q}}$ en comparant $|\mathscr{Q}(c)|$ à la valeur maximale de $|\mathscr{Q}(\bar{c})|$ où $\bar{c}$ est un successeur direct de c (ligne 6). Pour ce faire, la variable *new-max* est utilisée. Cette dernière désigne le concept dans $\mathscr{L}^+$ dont l'intension est égale à $\mathscr{Q}(c)$, c'est à dire, $\chi^+(c)$. Elle est initialisée avec le successeur $\bar{c}$ ayant la valeur maximale $|\mathscr{Q}(\bar{c})|$ (ligne 5). Les maximas de classes sont répartis en modifiés et géniteurs (ligne 7). Pour un concept modifié c (lignes 8 to 10), l'extension est mise à jour par l'objet o et est considérée comme étant son propre maxima (ligne 10) par l'intermédiaire de la variable *new-max* ($\chi^+(c) = c$). Un géniteur donne, d'abord naissance à un nouveau concept $\hat{c}$ (ligne 12) ensuite la valeur de χ^+ de chaque successeur direct est mise dans une liste locale (la liste *Candidates*, ligne 13) afin de pouvoir sélectionner les éléments minimas (MIN-CLOSED, ligne 14). Les concepts minimas sont liés au nouveau concept $\hat{c}$ (ligne 15) et ceux qui sont des modifiés dans $\mathscr{L}$ (ligne 16) sont déconnectés du géniteur c (ligne 17). Finalement, le maxima de la classe $[c]_{\mathscr{Q}}$

$c \in \mathscr{L}$	$\mathscr{Q}(c)$	*new-max*	Statut	$c \in \mathscr{L}^+$	$\chi^+(c)$
$c_{\#1}$	$\emptyset$	*null*	**M**(9)	$\{c_{\#1}\}$	$c_{\#1}$
$c_{\#2}$	c	*null*	**M**(9)	$\{c_{\#2}\}$	$c_{\#2}$
$c_{\#3}$	d	*null*	**M**(9)	$\{c_{\#3}\}$	$c_{\#3}$
$c_{\#4}$	g	*null*	**M**(9)	$\{c_{\#4}\}$	$c_{\#4}$
$c_{\#5}$	h	*null*	**M**(9)	$\{c_{\#5}\}$	$c_{\#5}$
$c_{\#6}$	c	$c_{\#2}$	**U**(9)	$\{c_{\#6}\}$	$c_{\#2}$
$c_{\#7}$	cd	*null*	**G**(9)	$\{c_{\#7}, c_{\#14}\}$	$c_{\#14}$
$c_{\#8}$	dgh	*null*	**M**(9)	$\{c_{\#8}\}$	$c_{\#8}$
$c_{\#9}$	fh	*null*	**G**(9)	$\{c_{\#9}, c_{\#15}\}$	$c_{\#9}$
$c_{\#10}$	cd	$c_{\#7}$	**U**(9)	$\{c_{\#10}\}$	$c_{\#10}$
$c_{\#11}$	$cdgh$	*null*	**G**(9)	$\{c_{\#11}, c_{\#16}\}$	$c_{\#11}$
$c_{\#12}$	$dfgh$	*null*	**G**(9)	$\{c_{\#12}, c_{\#17}\}$	$c_{\#12}$
$c_{\#13}$	$cdfgh$	*null*	**G**(9)	$\{c_{\#13}, c_{\#18}\}$	$c_{\#13}$

Tableau 2.4: Trace de l'exécution de l'algorithme 1 avec le treillis de la Figure 2.5 et le nouvel objet ($\{9\}$,$\{cdfgh\}$).

dans $\mathscr{L}^+$, c'est à dire, $\chi^+(c)$ est repositionné (ligne 18), le nouveau concept est inséré dans l'ensemble des concepts du treillis (ligne 19) et la structure *ChiPlus* qui représente χ^+ est mise à jour par le nouveau maxima de la classe $[c]_{\mathscr{Q}}$ pour une future utilisation (ligne 20).

Exemple : Soient le contexte de la Table 2.3 formé par $\mathscr{K} = (O = \{1,\ldots,8\}, A = \{a,\ldots,h\}, I)$ et le contexte $\mathscr{K}^+$ obtenu par l'ajout de la paire ($\{9\}$,$\{cdfgh\}$). La Table 2.4 illustre l'exécution de l'algorithme 1. Le treillis initial est donné par la Figure 2.5 tandis que le treillis résultat est donné par la Figure 2.6.

Pour illustrer la manière avec laquelle l'algorithme 1 traite un concept donné, considérons le cas du concept $c_{\#12} = (3, defgh)$. La valeur de $\mathscr{Q}(c_{\#12})$ est $dfgh$. Comme les successeurs directs de $c_{\#12}$ sont $c_{\#8}$ et $c_{\#9}$ (voir Figure 2.5), la liste des candidats contenant les images par χ^+ de ces successeurs est égale à : *Candidates*= $\{\underline{c}_{\#8} = (139, dgh), \underline{c}_{\#15} = (359, fh)\}$. Il est évident qu'aucune intension de successeur n'est plus grande que celle de $\mathscr{Q}(c_{\#12})$. Ainsi, $c_{\#12}$ est un maxima dont le statut est un géniteur. Le nouveau concept $\underline{c}_{\#17} = (39, dfgh)$ est

créé et connecté à tous ses successeurs directs dans *Candidates* car ils sont tous des éléments incomparables. Finalement, comme $\underline{c}_{\#8}$ est un modifié, le lien avec $c_{\#12}$ est supprimé.

2.2.2 INCRÉMENTALITÉ PAR ATTRIBUT

Une manière de construire le treillis $\mathscr{L}$ consiste à ajouter, progressivement, un attribut a_i au treillis $\mathscr{L}_{i-1}$ (qui correspond au contexte $\mathscr{K} = (O, \{a_1, ..., a_{i-1}\}, I)$), en commençant par le seul attribut a_1 et en faisant à chaque étape une série de mises à jour structurelles. Toutefois, cette façon de construire un treillis ne fait pas partie des objectifs de ce travail.

2.2.3 OUTILS DE CONSTRUCTION ET DE VISUALISATION DE TREILLIS

Plusieurs programmes sont disponibles permettant la construction de treillis. Le plus ancien (mais aussi parmi les plus fiables) d'entre eux est CONIMP qui s'exécute sous le système MSDOS et est accessible à l'adresse http ://www.mathematik.tu-darmstadt.de/ags/ag1/Software. Un autre programme plus moderne développé par *Navicon Company* pour les applications d'analyse de données est *Navicon Decision Suite* (TOSCANA, ANACONDA, CERNATO). Une version démo est accessible à l'adresse : http ://www.navicon.de. CONEXP est aussi un outil qui implémente toutes les fonctionnalités de base dans le domaines de l'AFC est disponible à l'adresse : http ://sourceforge.net/projects/conexp/. TOSCANAJ est aussi un explorateur de schémas conceptuels et est optimisé pour des novices en AFC et est disponible à l'adresse : http ://toscanaj.sourceforge.net/. Une autre plateforme de calcul de treillis appelée GALICIA (Valtchev et al., 2003b) supporte la totalité du cycle de vie d'un treillis. Une distribution gratuite de cette plateforme est accessible à l'adresse : https ://sourceforge.net/projects/galicia/.

2.3 QUELQUES APPLICATIONS USUELLES DE L'AFC

L'AFC est appliquée dans de nombreux domaines tels que l'analyse de données, le génie logiciel, la recherche d'information, l'extraction de connaissances, etc. Ces applications ont amplement contribué au développement théorique et algorithmique de l'AFC. Nous allons décrire quelques applications où les recherches sont très actives en ce moment. Une bonne partie de cette section est consacrée à la description de la manière avec laquelle l'AFC a abordé les problèmes du génie logiciel en particulier la réingénierie des systèmes et les solutions proposées pour améliorer leur qualité.

2.3.1 AFC ET EXTRACTION DE CONNAISSANCES

L'extraction de connaissances [10] (Piateski et Frawley, 1991) à partir de données (ECD) est une activité qui consiste à analyser les données afin de dégager de manière non triviale des informations implicites. À l'heure actuelle, ce domaine de recherche se trouve à la croisée de chemins de nombreux autres domaines tels que : systèmes de bases de données, intelligence artificielle, bases de données spatiales, visualisation de données, etc. Le processus d'ECD débute par l'analyse de données brutes et aboutit à des connaissances qui sont potentiellement utiles, dépendamment du type de données, à la gestion, le contrôle, la conception, la commercialisation, etc. Un processus classique d'ECD comprend le nettoyage et l'intégration de données à partir de diverses sources, la sélection et au besoin la transformation des données, la fouille de données [11] et finalement l'évaluation et la présentation des connaissances extraites. Une étape cruciale dans ce processus est la fouille de données dont le but est de chercher les patrons informatifs à partir des données et de les présenter sous une forme souhaitée

10. En Anglais knowledge discovery.
11. En Anglais data-mining.

telle que des règles d'implications ou une classification. C'est à ce moment qu'intervient l'AFC. En effet, l'AFC permet de représenter les données brutes par le biais de contextes formels et de dériver la hiérarchie conceptuelle correspondante (Duquenne, 1999). Des règles d'implication (Ganter et Wille, 1999) entre attributs peuvent être alors déduites de manière directe à partir du treillis permettant ainsi l'induction des hypothèses sur les relations entre attributs, ou d'extraire une classification des objets. L'avantage de l'utilisation de l'AFC avec le calcul des règles d'implication est la garantie d'obtenir une base de règles minimale (sans redondance des règles). De plus, les travaux sur l'incrémentalité des treillis ont conduit au développement d'algorithmes efficaces de mise à jour incrémentale d'une base de règles suite à l'évolution des données correspondantes.

2.3.1.1 Règles d'association et bases de règles

Le calcul des règles d'association (Agrawal et al., 1993) est considéré comme l'un des types de base d'extraction de connaissances dans les bases de données. Étant donnée une base de données, calculer les règles d'association consiste à déterminer toutes les règles qui ont un support et une confiance minimum. Une règle d'association est une implication de la forme $X \Rightarrow Y$, où X et Y sont des sous-ensembles de A, appelés aussi "*motifs*", et $X \cap Y = \emptyset$. Le support d'un ensemble X (X étant un ensemble d'attributs dans un contexte formel), noté $supp(X)$, est la proportion des transactions (les objets dans un contexte formel) contenant X. Le support d'une règle $X \Rightarrow Y$ est défini comme étant $supp(X \cup Y)$ alors que sa confidence est le rapport $supp(X \cup Y)/supp(X)$.

L'étape importante dans le calcul des règles d'association est la détermination des motifs fréquents à partir desquels les règles sont générées. Le nombre de motifs fréquents peut être potentiellement très grand conduisant à un nombre élevé de règles. Une approche basée sur

les opérateurs de fermeture a été proposée permettant de réduire le nombre des règles et par conséquent leur interprétation.

Ainsi, le calcul des motifs fréquent peut être ramené au calcul des motifs fermés fréquents [12] qui caractérise l'information pertinente d'un motif fréquent et qui conduit finalement à un ensemble de règles d'association minimal et représentatif, appelé "base". Plusieurs types de bases de règles d'association ont été proposés dans la littérature telles que la base de Duquenne-Guigues, la base de couverture de Luxemburger, la base générique et la base informative. Ces bases peuvent être produites à partir du treillis iceberg.

Les règles approximatives de couverture de Luxemburger sont générées à partir du motif fermé fréquent du concept et celui de ses prédécesseurs directs. La base de Duquenne-Guigues nécessite le calcul des pseudo-fermés fréquents associés au motif fermé fréquent d'un concept fréquent du treillis. La génération des règles exactes à partir de la base de Duquenne-Guigues est triviale car le pseudo-fermé devient une prémisse et le motif fermé fréquent du concept devient la conséquence de la règle.

Dans les bases génériques et informatives, les *générateurs* d'un motif fermé fréquent joue un rôle crucial dans le calcul des règles. Un générateur Z d'un ensemble fermé X est le sous-ensemble minimal de X tel que $Z'' = X$. Les règles exactes de la base générique sont produites à partir des concepts fréquents en utilisant leurs générateurs et les motifs fermés fréquents correspondants. Les règles approximatives de la base informative sont produites à partir des générateurs du concept fréquent et du motif fermé fréquent de ses prédécesseurs directs.

12. En Anglais Frequent Closed Itemsets (FCI).

2.3.2 AFC ET RECHERCHE D'INFORMATION

De nombreux travaux combinant AFC et recherche d'information ont montré le bien fondé de l'utilisation de l'AFC en recherche d'information (Carpineto et Romano, 1996). La correspondance entre les éléments de base de l'AFC et de la recherche d'information a été vite établie. En effet, les individus du contexte sont les objets recherchés (documents, utilisateurs, etc.) et les propriétés sont les caractéristiques qui leurs sont communes (mots-clés, liens hyper-documents, profiles, etc.). Un concept formel dans ce contexte représente un cluster d'individus ayant les mêmes propriétés, par exemple un ensemble de documents partageant un ensemble de mots-clés.

L'AFC permet de concilier les deux modes de recherche d'information : les systèmes hiérarchiques et les systèmes booléens. En effet, le treillis, grâce à la factorisation maximale des propriétés, offre une structure de recherche minimale qui combine les deux formes de recherche, à savoir, la navigation entre sous-population d'individus et l'interrogation (requête) de sous-ensembles de propriétés. L'exploration de la structure est garantie par les liens conceptuels offerts par le treillis.

Le principe d'exploration de la hiérarchie conceptuelle établie par les méthodes de l'AFC consiste à partir d'une requête vide qu'on raffine par ajouts successifs d'attributs (mots-clés). À chaque étape, on avance davantage dans la structure ce qui permet de restreindre l'espace de recherche, jusqu'à arriver à satisfaire la requête. On a donc une recherche par navigation dont le point de départ est le concept maximal du treillis (tous les objets) et le point d'arrivée est le concept possédant tous les attributs de la requête. L'extension de ce concept est rendue comme résultat de la recherche.

2.3.3 AFC ET RÉINGÉNIERIE DU LOGICIEL

Les systèmes légataires (Bennett, 1995; Lehman et Belady, 1985), devenus de plus en plus obsolètes dans leur architecture et/ou plateforme, présentent des difficultés à supporter l'évolution en fonction des changements des besoins. L'intérêt de la réingénierie (Arnold, 1992; K. H. Bennett et Robson, 1991; Cross et II, 1990) est l'amélioration ou la transformation de ces systèmes afin de les rendre plus compréhensibles, fiables, maintenables, assez documentés et facile à faire évoluer ou à migrer vers un autre paradigme. Les systèmes orientés objets (OO) contemporains peuvent aussi être soumis à une activité de réingénierie visant à corriger la conception ou à dégager un ensemble d'artéfacts potentiellement utiles à la maintenance. La réingénierie consiste donc à prendre un système existant et le reconstruire sous une autre forme en se basant sur un ensemble de processus élémentaires tels que la rétroingénierie, la redocumentation, la restructuration, etc. La rétroingénierie est le processus de compréhension et d'analyse d'un système dans le but d'identifier les composantes de celui-ci ainsi que leurs relations et de créer des représentations (abstractions) de ce système sous différentes formes ou à différents niveaux d'abstraction.

Depuis que Wille et Ganter ont développé l'AFC pour en faire une technique d'analyse et de dérivation de structures conceptuelles (Ganter et Wille, 1999; Wille, 1982), les applications ont proliféré, notamment en génie logiciel. En effet, grâce à sa faculté de regroupement et de structuration hiérarchique, l'AFC s'applique à des problèmes de la réingénierie (Arnold, 1992; K. H. Bennett et Robson, 1991; Cross et II, 1990) de logiciels où l'on s'intéresse à analyser le code existant pour des fins de compréhension et de maintenance.

Dans ce qui suit, nous allons décrire la manière avec laquelle l'AFC a abordé les problèmes de la réingénierie de logiciels orientés objet et les solutions proposées pour améliorer la qualité des systèmes.

2.3.3.1 Réingénierie des systèmes OO

De nos jours, on parle aussi de système légataires orientés objet qui souffrent eux aussi des mêmes symptômes que les systèmes légataires procéduraux. Des techniques de restructuration "refactoring" sont appliquées pour garder ces systèmes opérationnels. Cette restructuration est principalement dûe aux anomalies suivantes (Fowler et al., 1999) :

- usage impropre de l'héritage : la réutilisation du code versus polymorphisme,
- manque d'héritage : la duplication du code ou présence de l'instruction "switch",
- opérations mal placées : des opérations hors classes appropriées,
- violation de l'encapsulation : le transtypage d'instances et méthodes "friends".

Ainsi, pour remédier à ces anomalies dans la conception objet et à l'instar des systèmes procéduraux, les systèmes à objets ont "largement" bénéficié de l'apport de l'AFC comme une technique d'analyse et de structuration. En effet, plusieurs aspects de la conception objet ont été traités tels que la construction d'une hiérarchie de classes initiales à partir de la spécification des classes ou des objets, réingénierie des hiérarchies de classes existantes à partir des relations entre les classes et leurs attributs/méthodes membres, évolution de la hiérarchie de classes à partir des nouveaux besoins et l'assemblage de hiérarchies existantes.

Conception et réingénierie des hiérarchies de classes : Une activité majeure dans l'élaboration du modèle objet consiste à identifier les classes et les relations sémantiques les reliant. Les relations d'héritage représentent les liens sémantiques les plus sensibles du fait de l'influence majeure qu'elles ont sur la compréhension des modèles, l'intégrité sémantique des modèles, le potentiel de réutilisation des classes résultantes pour d'autres applications, et l'effort de développement requis.

Le problème de construction d'une hiérarchie de classes de départ à partir des spécifications

des différentes classes ou bien la réorganisation d'une hiérarchie existante a reçu une attention particulière dans la littérature des systèmes à objets. Du côté de l'AFC, le contexte formel est composé de classes et de membres de classes qui représentent, respectivement, les objets et attributs formels. La relation d'incidence, dans ce cas, représente pour chaque variable les membres de son type. Le treillis obtenu à partir de ce contexte comporte beaucoup de redondance. Ainsi, ce treillis subit plusieurs transformations qui consistent à éliminer les éléments redondants pour arriver à une structure qui n'est plus un treillis mais qui demeure un ordre partiel. Cette structure est la hiérarchie [13] de classes recherchée.

Travaux antérieurs : Godin et Mili (Godin et Valtchev, 2003; Godin et al., 1998) ont utilisé les treillis de Galois pour la réorganisation de la hiérarchie d'héritage en se basant sur la signature des classes. Le point de départ dans leur approche est un ensemble d'interfaces de classes. Une table binaire est construite représentant pour chaque interface l'ensemble des méthodes supportées. Le treillis dérivé à partir de cette table montre comment la hiérarchie de classes implémentant ces interfaces doit être organisée afin d'optimiser la répartition des méthodes dans la hiérarchie. Les travaux de Godin et Mili ont les mêmes bases formelles que ceux de Snelting (Snelting, 2000). Toutefois Godin et Mili considèrent les relations entre les membres et les classes alors que Snelting a étudié comment les membres d'une hiérarchie de classes sont utilisés dans le code exécutable de plusieurs applications en examinant non seulement les relations entre les classes et les membres de classes mais aussi les relations entres les différents membres de classes. Huchard et *al.* (Huchard et al., 2002) ont proposé une méthode de restructuration d'un modèle de classes UML permettant de considérer aussi bien les classes que les associations dans l'élaboration des généralisations.

13. Une hiérarchie de classes a seulement besoin d'être un ordre partiel.

2.3.4 DISCUSSION

Toute discipline dont les données peuvent être codées en un contexte formel (objets x attributs) peut bénéficier des avantages de l'AFC. En effet, l'AFC de par ses capacités de conceptualisation, structuration et factorisation apportent une solution élégante à chaque fois que les données brutes doivent être regroupées (concepts) selon des critères de similarité bien établis (partage objets/attributs) ou encore classifiées sans redondances (factorisation maximale). Toutefois, la taille des structures conceptuelles (treillis, iceberg, etc.) produites par les techniques de l'AFC pose un obstacle à l'exploitation de ces structures car elle dépend de manière exponentielle du volume de données. Ceci affaiblit la position de l'AFC face aux disciplines dont les ensembles de données (datasets) sont de grande taille.

2.4 CONCLUSION

Nous avons présenté les notions fondamentales de l'AFC, les structures conceptuelles qu'elle utilise et le mécanisme de scaling conceptuel qui lui permet de traiter les données non binaires. Dans le chapitre suivant sur l'état de l'art, nous présenterons une extension de l'AFC particulièrement intéressante car elle permet de prendre en charge des descriptions d'individus plus riches, notamment les liens inter-individus, connus aussi sous le nom de données relationnelles.

CHAPITRE 3

ÉTAT DE L'ART

Trouver un nom à un concept à partir d'une description est naturellement un problème de fouille et de conceptualisation. Ce problème a fait l'objet de nombreux travaux, principalement au sein de la communauté de fouille de textes (Miller, 1995; Suchanek et al., 2007; Gabrilovich et Markovitch, 2005; Egozi et al., 2008; Huang et al., 2009). Le problème a été abordé sous différents angles allant de la fouille de textes courts, à l'analyse de sentiments en passant par la classification/catégorisation, la détection d'un sujet de discours à partir du texte (Ang. topic mining), etc.

Dans ce qui suit, nous allons présenter quelques travaux pertinents dans ce sens. À noter que dans notre revue de la littérature, nous avons aussi exploré les ressources linguistiques (Miller, 1995) et ontologiques (Suchanek et al., 2007) qui pourraient servir à la découverte de concepts. Nous présenterons quelques unes des plus importantes.

3.1 FOUILLE DE TEXTES COURTS

La fouille de textes courts (p.e., des messages Twitter) considère le texte comme un ensemble de termes (Ang. bag of words). Plusieurs approches (Song et al., 2011; Gabrilovich et Mar-

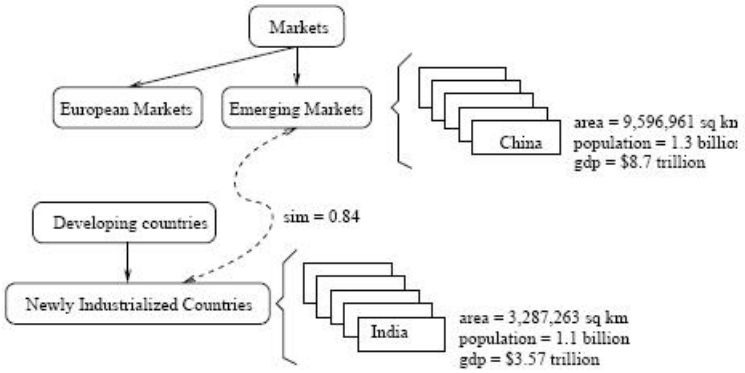


Figure 3.1: Un extrait de la taxonomie PROBASE.

kovitch, 2005; Egozi et al., 2008; Huang et al., 2009) ont été proposées. La majorité de ces approches sont basées sur une analyse statistiques et permettent d'identifier le sujet (Ang. topic) sur lequel porte l'ensemble des termes textuels. Par exemple, étant donnés les deux termes *Professeur* et *Étudiant*, la méthode de fouille devrait identifier le sujet '*Personne*'.

Étant donné que les termes ne possèdent pas assez de contenu pour pouvoir faire des inférences de tout genre (statistiques, probabilistes, etc.), certaines approches remplacent les termes par des artéfacts beaucoup plus riches obtenus à partir de ressources structurées tels que des articles Wikipedia (Gabrilovich et Markovitch, 2005; Egozi et al., 2008; Huang et al., 2009), bases de connaissances Probase (Song et al., 2011), etc.

Dans (Song et al., 2011), une approche pour la détection du sujet (Ang. topic modeling) est présentée. L'approche exploite la base de connaissances PROBASE [1] qui contient plus 2.7 millions de concepts obtenus à partir de l'analyse de 1.68 milliards de documents. PROBASE contient aussi des dizaines de millions de pairs concept-instance et terme-instance. L'inférence des relations entre les termes et les concepts associés est guidée par les instances (voir la figure 3.1).

En effet, étant donnée une description (un ensemble de termes), l'approche infère le concept

1. http ://research.microsoft.com/en-us/projects/probase/

associé dans PROBASE en appliquant un modèle probabiliste basé sur la règles de Bayes. D'abord, chaque terme de la description est mis en relation avec l'attribut/instance de PROBASE correspondant. Ensuite, le concept le plus probable est déterminé en utilisant les règles d'inférence de Bayes. Par exemple, étant donnée une description formée par les trois termes '*population*', '*language*' et '*currency*', le concept le plus probable proposé est '*country*'. L'approche a été appliquée pour la classification des messages Twitter.

3.2 CONCEPTUALISATION

Les approches proposées dans ce contexte focusent sur l'identification d'un concept pouvant servir à la classification/catégorisation. Nous nous sommes particulièrement intéressés aux approches de conceptualisation à partir d'un ensemble de mots (pour rester dans le cadre de notre problématique).

Dans (Hepp et al., 2007), les auteurs démontrent comment les URIs des articles de WIKIPEDIA (ou de toute autre technologie Wiki) peuvent être utilisés pour annoter des entités conceptuelles (dans notre cas les nouvelles abstractions fournies par la RCA) ou encore des ressources Web. La technique proposée est motivée par le fait que WIKIPEDIA est la plus grande collection d'éléments conceptuels et que ces éléments sont munis d'une description textuelle identifiable par des URIs. La technique a été appliquée avec succès à l'annotation des concepts de haut niveau de l'ontologie Proton [2].

2. http ://proton.semanticweb.org/

3.3 RESSOURCES LINGUISTIQUES/ONTOLOGIQUES PERTINENTES

3.3.1 WORDNET

Wordnet (Miller, 1995) est une base de données lexicale de la langue anglaise. Ces données sont structurées avec une unité de base appelée « synset », soit un ensemble de synonymes. Pour chaque mot répertorié, Wordnet fournit un ensemble de synset présentant les différents sens de ce mot.

Pour structurer ces unités de base, Wordnet utilise une panoplie de relations sémantiques et lexicales permettant de naviguer entre les synset. La structure est finalement assez proche d'une ontologie, chaque synset pouvant être assimilé à un concept. On pourra par exemple parcourir les relations d'hyperonymie (les relations de généralisation/spécialisation ou encore *est-un* entre concepts ; par exemple le concept Animal est l'hyperonyme du concept Mammifère) afin de trouver des concepts plus généraux, ou suivre la relation holonomie (qui permet d'établir des liaisons de type *partie-de* entre les différents concepts) et son inverse méronymie afin de naviguer entre un concept et ses parties. Ces deux relations seront particulièrement utiles dans le projet. Nous les verrons par la suite.

La base de données est consultable en ligne mais elle est aussi téléchargeable et consultable en local que ce soit avec l'application Wordnet ou avec les interfaces de programmation existantes, notamment l'API Java JWNL [3] permettant d'interroger Wordnet dans sa version installée en local. Toutes ces caractéristiques font de Wordnet une source de premier choix qui pourrait être exploitée pour la découverte d'un nom à partir d'une spécification de concept.

3. http ://sourceforge.net/projects/jwordnet/

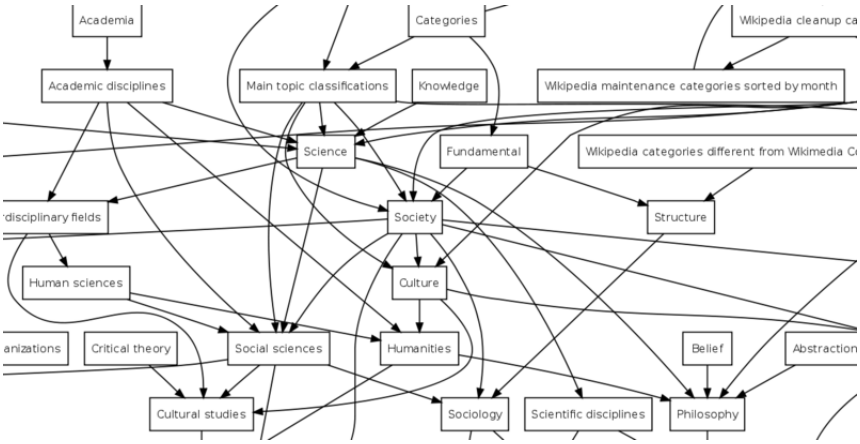


Figure 3.2: Une vue partielle du graphe des catégories Wikipedia. Les arcs sont orientés de la catégorie vers la sous-catégorie.

3.3.2 WIKIPEDIA

Wikipedia [4] est une encyclopédie universelle et multilingue développée par Wikimedia Foundation sous une licence libre [5]. De nos jours, elle constitue la plus grande base de connaissances.

Les articles de Wikipédia sont regroupés et rangés dans une ou plusieurs catégories. Les catégories sont, à leur tour, organisées (voir la figure 3.2) selon une hiérarchisation arborescente et thématique (par exemple « Pays d'Europe », puis « Italie », puis « Ville d'Italie ») [6].

Les entrées Wikipedia sont largement utilisées par les techniques d'apprentissage machine pour la fouille de texte (Milne et Witten, 2008; Hepp et al., 2007) ou encore la classification (Huang et al., 2009) de documents.

4. http ://en.wikipedia.org
5. Licence CC-BY-SA
6. http ://en.wikipedia.org/wiki/Wikipedia :Categorization

3.3.3 DBPEDIA

DBpedia (Auer et al., 2007) est un projet mené par l'université de Leipzig dont le but est d'extraire des données de Wikipédia et de les structurer sous la forme d'une ontologie.

DBpedia est une ressource très complète et pourra servir dans notre projet d'identification de noms appropriés pour les abstractions produites par le processus de restructuration d'une ontologie.

Dans (Stankovic et al., 2011), une technique est présentée permettant de recommander un sujet de discours en exploitant le graphe de concept de DBpedia.

3.4 RÉ-INGÉNIERIE DES ONTOLOGIES AVEC RCA

En représentation de connaissances, en particulier à l'aide d'une ontologie, la qualité de cette ontologie est encore une notion très peu formalisée. Néanmoins, deux facteurs très généraux peuvent être identifiés, chacun contribuant à augmenter l'utilisabilité d'un modèle ontologique : d'une part, la complétude du modèle, ou la présence de tous les concepts pertinents du domaine représenté, et d'autre part, l'absence de redondances dans celui-ci (qui, comme en bases de données, accroissent l'effort de maintenance). Bien qu'ils sont intuitifs, les deux critères sont loin d'admettre une satisfaction triviale et peuvent même s'avérer antinomiques si traités de façon indépendante. En revanche, il est très facile d'enfreindre les deux, surtout dans les modèles de grande taille ou à évolutions fréquentes. À titre d'illustration, examinons le diagramme de la Figure 3.3 qui reflète une partie d'une ontologie du domaine bancaire. En effet, les concepts représentent des types de comptes, soit chèque (`CheckAccount`) ou hypothèque (`MortgageAccount`), et leurs détenteurs respectifs (`CheckBookHolder` et `Mortgager`).

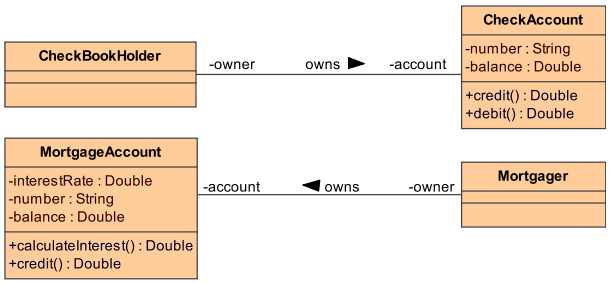


Figure 3.3: Portion d'un modèle statique du domaine bancaire.

Pris tel quel, le diagramme ci-dessus exhibe aussi bien des similarités que de l'incomplétude. En effet, le concept modélisant les deux types de comptes possèdent un bon nombre de propriétés en commun -Nous faisons l'hypothèse que les éléments de même nom ont une sémantique identique qui auraient été avantageusement factorisée par un super-concept, `Account` qui représente un compte bancaire. Le traitement de ce type de situations, fréquentes en construction, intégration ou maintenance de modèles, a été abordé avec des approches semi-automatiques, basées pour la majorité sur l'analyse formelle de concepts (AFC) Godin et al. (1998); Dao et al. (2004); Snelting (2000). En effet, AFC s'est avérée être un cadre théorique adéquat pour la restructuration des hiérarchies de par la proximité entre la structure d'une hiérarchie de concepts et celle d'un treillis de concepts formels, ainsi que par la factorisation maximale que les treillis et les structures de concepts dérivées assurent aux hiérarchies qui en sont extraites. Cependant, l'AFC classique a ses limites. Pour s'en persuader, il suffit d'observer l'analogie des rapports que les concepts des détenteurs entretiennent avec les concepts comptes respectifs sur la Figure 3.3. Ainsi, un super-concept `Client` semble être la réponse appropriée qui de plus permettra de factoriser le rôle `owns`. Toutefois, `CheckBookHolder` et `Mortgager` n'ont aucun membre en commun, alors qu'aucune technique actuelle d'AFC ne permet de regrouper les individus sur la base d'une analogie dans leurs liens à d'autres individus.

Dans la suite de cette étude de l'état de l'art, un cadre plus expressif pour l'extraction de

concepts formels sera présenté et les divers problèmes concrets soulevés par son application à la restructuration des modèles ontologiques seront abordés.

3.4.1 LIMITES DE L'ANALYSE FORMELLE DE CONCEPTS

La table illustrée par la Figure 3.4 montre un contexte binaire $\mathscr{K} = (O, A, I)$ où les individus sont les concepts de l'ontologie de la Figure 3.3 et les attributs formels sont les propriétés de ces concepts.

	number (nb)	balance (ba)	interestRate (ir)	credit() (cr)	debit() (de)	calculateInterest() (ci)
CheckBookHolder (CBH)						
CheckAccount (CA)	X	X		X	X	
Mortgager (M)						
MortgageAccount (MA)	X	X	X	X		X

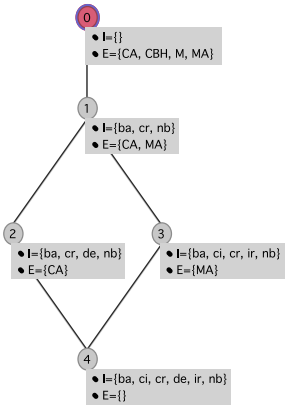


Figure 3.4: Gauche : Codage des concepts ontologiques de la Figure 3.3 par un contexte binaire. Droite : Le treillis de concepts formels correspondant.

La Figure 3.4 illustre un contexte et le treillis de concepts correspondant. Les concepts du treillis représentent toutes les factorisations maximales des propriétés des individus. Le concept formel $c_{\#1}$, par exemple, est composé de l'extension {`CA`,`MA`} et de l'intension {`ba`, `cr`, `nb`}. Cela signifie que les deux concepts ontologiques `CheckAccount` et `MortgageAccount` partagent les variables `number`, `balance` et l'opération `credit`.

L'AFC de par la définition formelle de la notion de « concept », de « hiérarchie conceptuelle » et de son pouvoir d'abstraction et de factorisation maximale s'adapte très bien au problème d'analyse Snelting (2000) et de construction Godin et al. (1998); Huchard et al. (2000) de

hiérarchies d'abstractions ainsi que la découverte de patrons de conception Tonella et Antoniol (1999). Le concept en AFC est la représentation formelle d'un concept physique ou abstrait du domaine (concept métier). La hiérarchie conceptuelle précise les relations de spécialisation/généralisation entre concepts. L'abstraction permet de former de nouveaux concepts (super-concepts) à partir des traits communs d'un ensemble d'individus (propriétés/rôles). Enfin, la factorisation maximale permet d'avoir une structure sans redondances et favorise par conséquent la réutilisation.

Bien que les techniques de l'AFC ont été appliquées avec succès pour la maintenance des logiciels objets (modèles UML), la richesse des données rencontrées dans les ontologies, en fort contraste avec la relative simplicité du modèle binaire de données utilisées en AFC, a révélé les limites du paradigme « classique » de l'AFC. Par exemple, le treillis de la Figure 3.4, nous a permis de détecter une seule abstraction possible dans le modèle de la Figure 3.3, le concept `Account`, alors que d'autres auraient été possibles. En effet, `CheckBookHolder` et `Mortgager` jouent un rôle analogue par rapport aux concepts ontologiques qui correspondent aux comptes. Intuitivement, elles pourront être abstraites vers un super-concept commun qu'on appellerait `Client` et qui factoriserait leur rôle de détenteurs de compte. En se penchant sur les éléments du modèle qui pourraient servir de base pour cette dernière abstraction, on constate que `CheckAccount` et `MortgageAccount` participent à des rôles ayant une sémantique proche les reliant à des sous-concepts de `Client`. Donc, deux questions surviennent naturellement : Comment reconnaître la proximité des sémantiques « proche » ? et comment signaler l'existence du super-concept de `CheckBookHolder` et `Mortgager`, soit `Client`, vers `CheckAccount` et `MortgageAccount`. La première question appelle clairement à un traitement linguistique (par exemple, une analyse des sens des deux termes). La seconde sort du cadre actuel de l'AFC en requérant de nouveaux mécanismes d'abstraction capables de regrouper les individus non seulement en fonction du partage dans leurs propres descriptions,

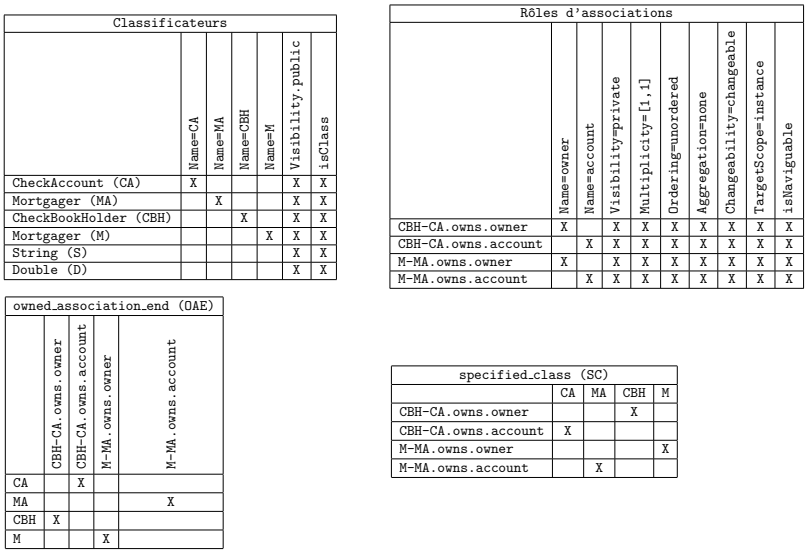
mais aussi dans les descriptions des individus qui leurs sont liés. De plus, le modèle devrait pouvoir être analysé en profondeur, c'est-à-dire, ne pas se limiter aux abstractions de concepts mais étendre l'analyse à d'autres types d'éléments tels que les attributs, relations sémantiques, etc. Cependant, les liens existant entre ces éléments comme l'incidence concept-attribut, le typage d'un rôle par un concept, etc. constituent des informations *relationnelles* qui ne sont pas à la portée des approches d'AFC actuelles. L'obstacle devant les méthodes automatiques capables de produire des abstractions de cette complexité est de taille car il s'agit de modifier la façon même dont les concepts sont construits en y intégrant le partage de liens à côté de celui des attributs formels.

3.4.2 ANALYSE RELATIONNELLE DE CONCEPTS

En ARC, les données sont décrites par une collection de contextes et de relations binaires appelée famille de contextes relationnels (FCR). La Figure 3.5 illustre une partie de la FCR extraite du modèle ontologique de la Figure 3.3 par la façon décrite dans la section 3.4.3.

Définition 24 *Une FCR est une paire* $(\mathbf{K},\mathbf{R})$ *où* $\mathbf{K}$ *est un ensemble de contextes pluri-valués* $\mathscr{K}_i = (O_i, A_i, V_i, I_i)$ *et* $\mathbf{R}$ *est un ensemble de relations* $r_k \subseteq O_i \times O_j$ *où* O_i *et* O_j *sont des ensembles d'individus, appelés domaine et co-domaine de* r_k*, respectivement.*

Les domaines de valeurs des relations inter-contextes sont des sous-ensembles d'individus. Ainsi, à l'instar du scaling conceptuel en AFC Ganter et Wille (2001), le scaling relationnel traduit les structures de ces domaines par des prédicats décrivant les sous-ensembles d'individus. Nous avons choisi de baser ces prédicats sur la structure conceptuelle de l'ensemble des individus référencés. Ainsi, les nouveaux attributs formels issus du scaling —ayant la forme $r : c$ où r et c désignent un nom d'une relation et d'un concept formel, respectivement— reflèteront



Classificateurs

	Name=CA	Name=MA	Name=CBH	Name=M	Visibility.public	isClass
CheckAccount (CA)	X				X	X
Mortgager (MA)		X			X	X
CheckBookHolder (CBH)			X		X	X
Mortgager (M)				X	X	X
String (S)					X	X
Double (D)					X	X

Rôles d'associations

	Name=owner	Name=account	Visibility=private	Multiplicity=[1,1]	Ordering=unordered	Aggregation=none	Changeability=changeable	TargetScope=instance	isNavigable
CBH-CA.owns.owner	X		X	X	X	X	X	X	X
CBH-CA.owns.account		X	X	X	X	X	X	X	X
M-MA.owns.owner	X		X	X	X	X	X	X	X
M-MA.owns.account		X	X	X	X	X	X	X	X

owned_association_end (OAE)

	CBH-CA.owns.owner	CBH-CA.owns.account	M-MA.owns.owner	M-MA.owns.account
CA		X		
MA				X
CBH	X			
M			X	

specified_class (SC)

	CA	MA	CBH	M
CBH-CA.owns.owner			X	
CBH-CA.owns.account	X			
M-MA.owns.owner				X
M-MA.owns.account		X		

Figure 3.5: Une famille de contextes relationnels codant les concepts et les relations sémantiques de l'ontologie de la Figure 3.3 ainsi que leurs liens.

l'appartenance d'un individu de l'ensemble à analyser à un concept identifié au préalable. Par exemple, si un concept formel représentant les propriétés de données de type réel est identifié, un individu du contexte des concepts ontologiques pourra se voir affecter un attribut formel signalant la possession d'au moins une telle donnée ontologique. Formellement :

Définition 25 *Étant donné un contexte $\mathscr{K}_i = (O_i, A_i, I_i)$, le treillis de concepts $\mathscr{L}_j$ associé au contexte $\mathscr{K}_j = (O_j, A_j, I_j)$ et la relation $r \subseteq O_i \times O_j$, l'opérateur de scaling $sc_{\times}^{(r,\mathscr{L}_j)} : \mathbf{K} \to \mathbf{K}$ est défini par : $sc_{\times}^{(r,\mathscr{L}_j)}(\mathscr{K}_i) = (O_i^{(r,\mathscr{L}_j)}, A_i^{(r,\mathscr{L}_j)}, I_i^{(r,\mathscr{L}_j)})$, où : $O_i^{(r,\mathscr{L}_j)} = O_i$, $A_i^{(r,\mathscr{L}_j)} = A_i \cup \{r : c | c \in \mathscr{L}_j\}$, $I_i^{(r,\mathscr{L}_j)} = I_i \cup \{(o, r : c) | o \in O_i, c = (X,Y) \in \mathscr{L}_j, r(o) \neq \emptyset, r(o) \subseteq X\}$).*

L'opérateur $sc_{\times}^{(r,\mathscr{L}_j)}$ est dit « *étroit* » car la manière de coder la relation d'incidence entre un individu o de l'ensemble à analyser O_i et les nouveaux attributs formels de la forme $r : c$ repose sur l'opération d'inclusion $\subseteq$ qui exige que tous les individus désignés par les liens de type r en partance de o (l'ensemble $r(o)$) soient inclus dans l'extension du concept c. Par contre, pour l'opérateur de scaling dit « *large* », utilisé pour un modèle ontologique et noté

$sc_+^{(r,\mathscr{L}_j)}$, il suffit qu'il y ait des individus désignés par les liens de type r dans l'extension de c. Ainsi pour $sc_+^{(r,\mathscr{L}_j)}$, $I_i^{(r,\mathscr{L}_j)} = I_i \cup \{(o, r:c) | o \in O_i, c = (X,Y) \in \mathscr{L}_j, r(o) \cap X \neq \emptyset\})$.

Par exemple, le scaling de la relation `Owned_Association_End` (`OAE`) reliant les concepts ontologiques à leurs rôles respectifs (voir la Figure 3.5) étend le contexte des classificateurs par les nouveaux attributs formels `OAE:c0` et `OAE:c3` où `c0` et `c3` sont des concepts formels du treillis dérivé à partir du contexte des rôles de la Figure 3.5 et correspondent aux rôles `owner` et `account` de la Figure 3.3, respectivement. L'objet formel `CA` (Concept ontologique `CheckAccount`), par exemple, a l'attribut formel `OAE:c3` car le concept ontologique `CheckAccount` possède le rôle `account` comme illustré sur la Figure 3.3.

Le scaling relationnel est au cœur d'un processus qui, à partir d'une FCR construit une famille de treillis relationnels (FTR), un treillis par contexte. Au sein d'une FTR, les concepts renferment dans leurs extensions des individus qui, en plus des caractéristiques de départ, partagent aussi des liens vers d'autres individus. Ce nouveau partage se matérialise à travers des attributs formels appelés « *relationnels* ». Tout attribut relationnel peut être interprété par une relation entre deux concepts, d'une part un concept représentant un groupe d'individus et d'autre part un concept représentant le groupe d'individus référencé par un même type de lien. À noter que le processus d'extraction des treillis est de nature itérative car le scaling relationnel modifie l'état des contextes ce qui nécessite la mise à jour des treillis associés qui à son tour implique un re-scaling de toutes les relations ayant utilisé les treillis qui ont été modifiés comme source de concepts pour la fabrication de prédicats décrivant des individus référencés. Le processus itératif de construction de la FTR s'arrête dès qu'un point fixe est atteint, c'est-à-dire lorsqu'un nouveau scaling de toutes les relations de la FCR ne conduit à l'enrichissement d'aucun contexte.

La Figure 3.6 et la Figure 3.7 illustrent le treillis final du contexte des concepts ontologiques

(classificateurs) et du contexte des rôles de la Figure 3.5, respectivement.

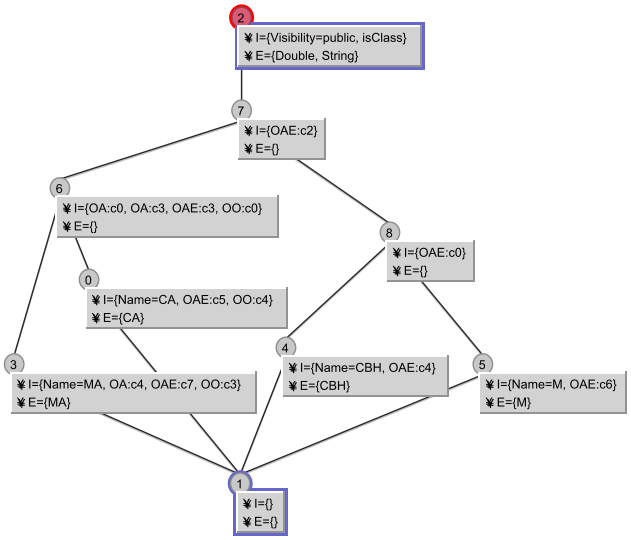


Figure 3.6: Le treillis de concepts formels final du contexte des concepts ontologiques (classificateurs).

Par exemple, le concept $c_{\#5} = (\{M\}, \{name{=}M, OAE : c6\})$ du treillis des concepts ontologiques représente le concept `Mortgager` de la Figure 3.3. L'attribut relationnel $OAE : c6$ indique que le concept formel $c_{\#5}$ du treillis des concepts ontologiques (Figure 3.6) a une relation de type *OAE* avec le concept formel $c_{\#6}$ du treillis des rôles (Figure 3.7). Ceci signifie que le concept ontologique `Mortgager` possède le rôle `owner` (voir Figure 3.3).

3.4.3 ARC ET RESTRUCTURATION DES ONTOLOGIES

3.4.3.1 Transformation d'une ontologie vers une FCR

L'ontologie est codée par plusieurs contextes, un par type d'élément dans l'ontologie qui sera sujet à un processus d'abstraction (typiquement, concepts et rôles) ; puis ces contextes sont reliés par le biais de méta-associations, appelées *techniques*. Ces associations ont une sémantique indépendante du domaine sous-jacent et traduisent les liens qui existent entre

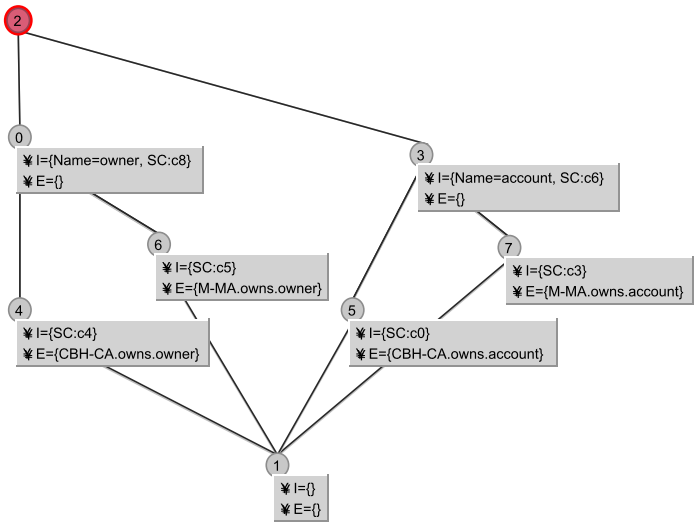


Figure 3.7: Le treillis de concepts formel final du contexte des rôles ontologiques.

éléments au sein de l'ontologie. L'avantage d'un tel codage est la possibilité d'abstraire sur plusieurs types d'éléments et d'opérer des enrichissements mutuels entre les hiérarchies d'abstractions sous-jacentes.

La liste complète des contextes et des relations techniques a été établie suite à une étude du méta-modèle ontologique. Ceux-ci sont illustrés sur la Figure 3.8, où certaines relations ont dû être nommées de façon plus explicite. En général, les objets formels de chaque contexte représentent des instances du même méta-concept. Les attributs formels dans les différents contextes représentent les propriétés locales aux individus, c'est-à-dire, des méta-attributs. Toutefois, tous les méta-attributs ne sont pas nécessairement pertinents pour nos objectifs et certains peuvent même nuire à l'abstraction. Ainsi, il est inutile de chercher à factoriser des propriétés comme `isLeaf` ou `isRoot` car leur valeurs définitives ne peuvent que se déterminer une fois la restructuration terminée. La Figure 3.5 montre une manière de coder les concepts et les rôles du modèle ontologique de la Figure 3.3.

Parmi les difficultés que rencontre l'application systématique de l'ARC, il y a l'apparente

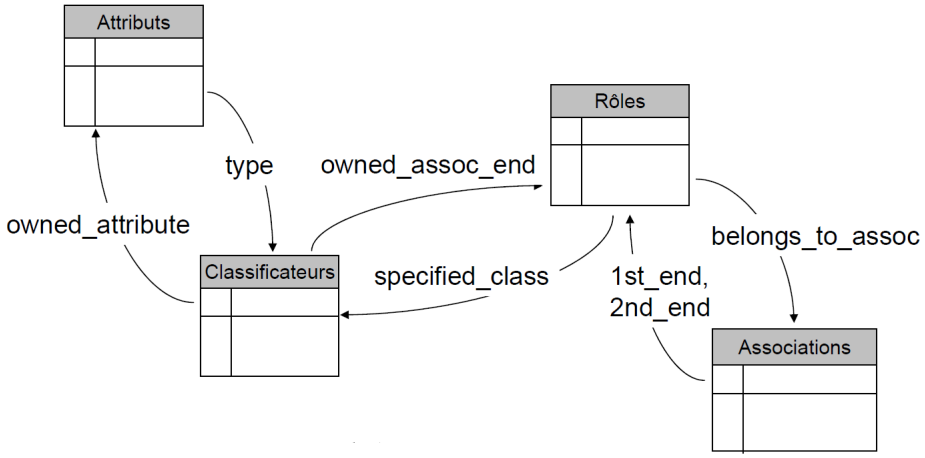


Figure 3.8: Le schéma générique de la FCR d'une ontologie.

antinomie entre la nécessité de préserver toutes les informations incluses dans le modèle ontologique initial, d'une part, et le besoin de réduire les abstractions fortuites, d'autre part. Ainsi, un modèle de taille raisonnable peut engendrer un grand nombre d'abstractions à faible utilité. En outre, un facteur de bruit important réside dans la polysémie du langage naturel qui est utilisé pour nommer les éléments du modèle. Par exemple, il est possible de trouver dans les grands modèles des éléments dont le nom est différent mais la sémantique est identique et inversement. Ces problèmes ont été considérés dans le cadre de l'intégration de schémas de base de données et la solution standard habituellement suggérée était de résoudre les conflits de noms par une normalisation des noms, en se servant de ressources linguistiques telles que des dictionnaires électroniques ou des ontologies de domaine Rahm et Bernstein (2001).

3.4.3.2 Génération d'une ontologie à partir d'une FTR

La génération d'une ontologie à partir d'une FTR est une opération complexe car, d'une part, les treillis peuvent être de taille exponentielle par rapport au nombre d'éléments du modèle ontologique initial et d'autre part, les relations inter-treillis peuvent être de nature circulaire

et/ou transitive. La méthode est basée sur un processus exploratoire et semi-automatique faisant intervenir des outils de navigation de treillis et le concepteur pour préciser des choix de modélisation selon ses objectifs. La méthode se déroule en trois étapes principales :

— Sélection des concepts du treillis des classificateurs (concepts-classificateurs) de départ : cette étape consiste à déterminer les concepts-classificateurs qui correspondent aux concepts ontologiques du modèle initial. Ces concepts, une fois traduits en langage ontologique, représentent forcément des concepts métiers. Par exemple, les concepts du treillis final des concepts ontologiques illustrés par la Figure 3.6 représentant les concepts du modèle initial sont : $c_{\#0}$, $c_{\#3}$, $c_{\#4}$ et $c_{\#5}$ et correspondent aux concepts ontologiques `CheckAccount`, `MortageAccount`, `CheckBookHolder` et `Mortager`, respectivement.

— Détection des abstractions significatives en parcourant les concepts-classificateurs retenus à l'étape précédente et en cherchant parmi leurs successeurs immédiats (Cov^u) les autres pertinents. Les concepts pertinents sont des concepts dont l'intension contient des informations métiers, soit directement, c'est-à-dire, à travers les attributs formels comme *nom*=..., *type*=..., etc. exprimant la factorisation d'une information du domaine, soit indirectement, par la référence relationnelle vers un autre concept, éventuellement sur un autre type d'élément du modèle, qui a été reconnu comme pertinent auparavant. Par exemple, les concepts formels successeurs des concepts retenus lors de la première étape sont : $c_{\#6}$, $c_{\#7}$, $c_{\#8}$ et $c_{\#2}$ dont seuls $c_{\#6}$ et $c_{\#8}$ sont pertinents.

— Génération du modèle ontologique par la traduction en éléments ontologiques des concepts formels du treillis des concepts ontologiques retenus et des concepts des autres treillis qu'ils désignent. Ainsi, les concepts formels retenus sont traduits en concepts ontologiques, les liens d'héritage sont déduits de la relation d'ordre entre eux et l'intension de chaque concept-classificateur retenu est traduite en éléments de

modélisation tels que attributs, rôles, opérations, etc. en exploitant les informations fournies par les concepts désignés dans les différents attributs relationnels.

3.4.3.3 Nommage des nouvelles abstractions

il est nécessaire de donner des noms aux éléments du modèle restructuré. Notre travail consiste à développer une approche de nommage efficace qui sera présentée dans le chapitre à venir.

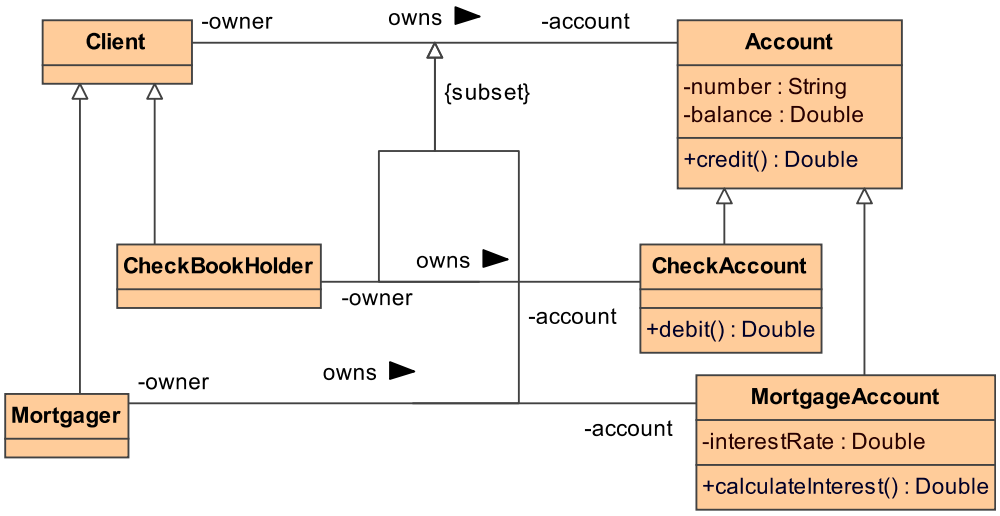


Figure 3.9: L'ontologie finale obtenue après la restructuration de l'ontologie de la Figure 3.3.

La Figure 3.9 montre la traduction en éléments ontologiques de tous les concepts formels pertinents dans les treillis de concepts finaux. Comparativement au modèle initial de la Figure 3.3, le nouveau modèle présente un meilleur niveau d'abstraction et de factorisation. On constate l'émergence de deux nouveaux concepts clés dans le domaine bancaire, à savoir, `Client` et `Account` avec un nouveau rôle les reliant qui généralise les liens entre les concepts dérivés.

3.5 CONCLUSION

Nous avons conduit une revue de la littérature pour, d'abord identifier le cadre dans lequel s'inscrit notre problème, ensuite explorer les solutions qui ont été proposées.

Nous avons réalisé que le problème d'affectation d'un nom à une abstraction nouvellement créée dans le cadre de l'enrichissement d'une ontologie existante a été abordé dans la littérature sous l'angle de la fouille de textes courts. En effet, l'abstraction su-citée est fournie par le mécanisme de l'ARC et est décrite par deux ensembles de termes appelés intension et extension, respectivement. De nombreux travaux ont été effectués dans ce sens et plusieurs approches sont proposées. La majorité de ces approches exploitent des ressources linguistiques et/ou ontologiques existantes vu la pauvreté lexicale et l'absence de sémantique dans un ensemble de terme face aux techniques de fouille de texte.

CHAPITRE 4

APPROCHE DE FOUILLE D'ANNOTATIONS DE CONCEPTS

4.1 INTRODUCTION

Nous partons d'une vision pour la solution du problème de fouille de concepts qui est fortement liée à l'approche de restructuration utilisée dans INUKHUK. Cette approche, qui est basée sur le cadre formel de l'analyse relationnelle de concepts (RCA), prend en entrée un ensemble de concepts ontologiques avec leurs descriptions associées (p.e., les propriétés) et produit en sortie de nouveaux concepts candidats qui reflètent tous les partages pertinents dans les descriptions. La description des nouveaux concepts est composée de deux ensembles. D'abord, l'extension, un ensemble de termes qui correspondent à des noms de concepts ontologiques. Ensuite, l'intension, un ensemble de termes qui correspondent à des propriétés partagées par les concepts ontologiques présents dans l'extension. Ainsi, pour chaque concept candidat, nous disposons de deux descriptions pouvant être utilisées de façon séparés ou combinée pour trouver un nom approprié à ce candidat. Les noms candidats issus de l'analyse des deux descriptions peuvent être confrontés à titre de validation. Dans ce chapitre, nous allons proposer une approche originale permettant d'affecter un nom, dit *candidat* à un concept, dit *inconnu*, à partir d'une courte description (l'intension et/ou l'extension) de celui-ci.

4.2 MÉTHODE DE FOUILLE PROPOSÉE

Comme nous disposons, pour tout concept inconnu, de deux descriptions, à savoir l'extension et l'intension, nous nous proposons donc de mener deux types de fouille d'un nom approprié pour le concept inconnu. Les deux méthodes s'appuient sur une source de connaissances existante (p.ex., une ressource ontologique). Vu la diversité des sources de connaissances et des objectifs respectifs, nous choisirons la source appropriée pour chaque type de fouille à l'étude expérimentale. En effet, à l'étape de l'expérimentation, nous allons étudier l'adéquation de plusieurs sources de connaissances (WIKIPEDIA, PROBASE, WORDNET, DBPEDIA, etc.) pour chaque type de fouille et nous mesurerons la qualité des résultats fournis en terme de précision et de rappel.

4.2.1 DESCRIPTION DU PROCESSUS DE FOUILLE

La Figure 4.1 illustre le processus général de fouille proposé. Le processus réalise, d'abord, deux types de fouilles. La première fouille (Extent-driven miner) démarre avec une liste de termes (des noms de concepts ontologiques) qui représentent l'extension du concept que nous cherchons à nommer et une ressource ontologique qui représente une source de connaissances permettant de faire des généralisations sur les termes en entrée. La seconde fouille (Intent-driven miner) démarre avec un ensemble de termes (propriétés de concepts ontologiques) qui représentent l'intension du concept que nous cherchons à nommer. À l'instar de la première fouille, la seconde fouille prend aussi en entrée une source de connaissances existante lui permettant d'identifier un concept dont la description correspond à l'ensemble des termes en entrée. Les résultats des deux fouilles sont ensuite confrontés dans le cadre d'un processus de croisement (cross miner) afin de fournir une liste unique de candidats pour le concept à nommer.

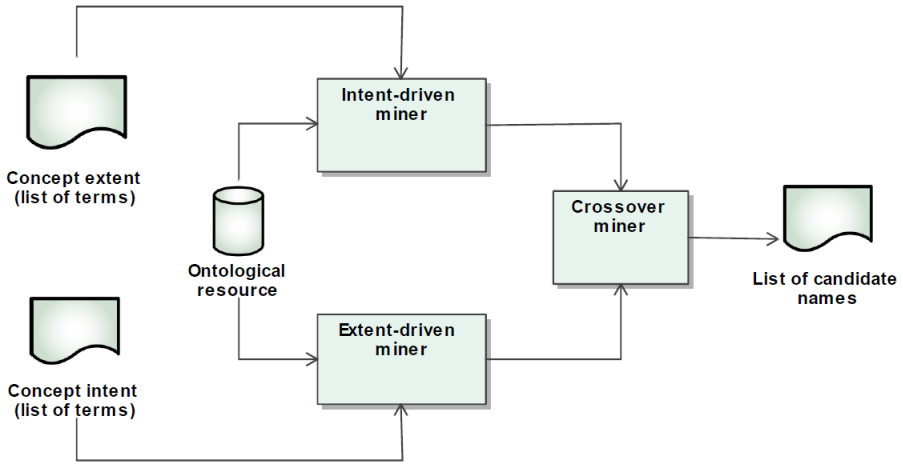


Figure 4.1: Vue générale de l'approche de fouille de noms de TOPICMINER.

4.2.2 ALGORITHME GÉNÉRAL DE FOUILLE

La méthode de fouille illustrée par l'algorithme 2 permet de retourner une liste de noms possibles, ordonnés par pertinence, pour un concept inconnu dont la description est donnée en entrée. La méthode prend aussi en entrée une source de connaissances (ligne 3). L'algorithme parcourt l'ensemble des termes qui composent la description (ligne 9). Pour chaque terme de cet ensemble, l'algorithme cherche le concept (dit dérivé) correspondant dans la source de connaissances fournies (ligne 10). Une fois que tous les concepts dérivés sont calculés, l'algorithme calcule les concepts généraux qui peuvent être déduits à partir des concepts dérivés (la ligne 12). Idéalement, un seul concept général devrait être identifié. Ce concept sera associé à la description fournie en entrée. La méthode COMPUTE-GENERAL-CONCEPTS est implémentée en fonction de la structure de la source de connaissances. Ainsi, chaque source donne lieu à une instance différente de cette méthode. Dans la section 4.3, nous allons donner quelques instances pour les sources de connaissances les plus usuelles tels que WIKIPEDIA.

```
 1: Routine DESCRIPTION-DRIVEN-MINING
 2: Input : Set ℰ of terms t_i that represent the extent of the unknown concept
 3: Input : Ontological resource ℛ
 4: Local : List of concepts 𝒞
 5: Output : Ordered list 𝒩 of names n_j for the unknown concept
 6:
 7: BEGIN
 8: 𝒞 = ∅
 9: for all t_i in ℰ do
10:    c_i = RETRIEVE-CONCEPT(ℛ,t_i)
11:    𝒞 = 𝒞 ∪ { c_i }
12: 𝒩 = COMPUTE-GENERAL-CONCEPTS(𝒞)
13: Return 𝒩
14: END
```

Algorithm 2: Traitement d'une description avec une source de connaissances.

Fouille dirigée par l'extension : L'extension comme mentionnée ci-haut est un ensemble de termes qui correspondent à des noms de concepts ontologiques connus. De point de vue de la conceptualisation, le concept inconnu associé à cette extension devrait être la généralisation des concepts connus de cet ensemble. De ce fait, nous proposons de conduire une recherche d'une généralisation appropriée d'un ensemble de concepts dans les ressources ontologiques à la disposition de la communauté de recherche.

Fouille dirigée par l'intension : L'intension comme mentionnée au début de ce chapitre est un ensemble de propriétés de concepts ontologiques. De point de vue de la conceptualisation, le concept inconnu associé à cette intension devrait être caractérisé par chacune des ces propriétés. Par conséquent, nous proposons de lancer une recherche d'un concept adéquat dans une ontologie lexicale qui renferme beaucoup de termes servant à décrire des concepts ontologiques.

4.2.3 FOUILLE DE CROISEMENT

Le but ultime de la fouille de croisement est d'améliorer les performances de l'approche proposée pour la fouille d'un nom approprié d'un concept inconnu. Ainsi, on se propose de mener deux fouilles de types différents, par exemple, une fouille dirigée par les extensions et une autre dirigée par les intensions, ou encore deux fouilles d'un même type, mais avec deux sources de connaissances distinctes. Nous sommes convaincu que la confrontation des résultats de deux fouilles permet de produire une liste filtrée suffisamment restreinte dont les éléments (des noms candidats) sont plus pertinents. De plus, afin d'affiner davantage la liste des candidats de la fouille de croisement, nous proposons d'utiliser le voisinage du concept inconnu. En effet, les concepts voisins, s'ils sont connus, font partie du même domaine sémantique que le concept inconnu. De ce fait, on peut filtrer, ou avantager certains candidats de la liste de croisement par rapport à d'autres candidats en utilisant l'information de voisinage.

Description de l'algorithme de croisement : La méthode de fouille par croisement prend en entrée deux ensembles de noms possibles (lignes $2-3$) ainsi qu'un seuil (ligne 4) à partir duquel les résultats du calcul sont acceptés ou refusés. La méthode de croisement consiste à calculer une matrice de distance [1] sémantique entre les termes issues des deux ensembles (lignes $8-10$). Les distances qui passent le filtre (ligne 11) sont acceptées et les termes respectifs issues des deux ensembles en entrée sont combinés pour obtenir un terme unique. Pour le moment la combinaison consiste à choisir de façon aléatoire un terme parmi les deux. Toutefois, nous nous proposons de faire une étude approfondie de cette question dans les futurs travaux. Par la suite, le terme retenue est placé dans l'ensemble à produire en sortie (ligne 13). Au final, l'ensemble de sortie est trié selon la distance associée (ligne 14).

1. En utilisant l'API http ://code.google.com/p/ws4j/

```
 1: Routine CROSSOVER-MINING
 2: Input : Set 𝒩1 of names n1i that represent the names provided by extent-driven miner
 3: Input : Set 𝒩2 of names n2j that represent the names provided by intent-driven miner
 4: Input : A threshold α
 5: Output : Ordered list 𝒩 of names nk for the unknown concept
 6:
 7: BEGIN
 8: for all n1i in 𝒩1 do
 9:    for all n2j in 𝒩2 do
10:       dij = COMPUTE-DISTANCE(n1i,n2j)
11:       if (dij ≤ α) then
12:          nij = DERIVE-UNIQUE-NAME(n1i,n2j)
13:          UPDATE-LIST(𝒩,nij,dij)
14: SORT-LIST(𝒩)
15: Return 𝒩
16: END
```

Algorithm 3: Croisement des résultats de deux fouilles distinctes, p.ex., la fouille dirigée par les extensions et celle dirigée par les intensions.

4.3 RESSOURCES DE FOUILLE

À l'heure actuelle, il est difficile d'envisager des ressources ontologiques riches, en maintenance continuelle et ayant un accès libre au public et aux programmes via une API autre que WORDNET, WIKIPEDIA et les projets sous-jacents notamment DBPEDIA.

L'ontologie WIKIPEDIA semble bien appropriée à la fouille dirigée par l'extension. En effet, au sein de WIKIPEDIA, les articles (des termes) qui couvent un même sujet sont placés sous la même catégorie. De ce fait, la structure catégorique de WIKIPEDIA permet de retrouver, pour un ensemble d'articles (dans notre cas, des noms de concepts ontologiques connus), une catégorie minimale commune (Ang. Lowest Common Submumer (LCS)) qui pourrait être considérée comme étant le thème général de ces articles et par conséquent utilisée pour nommer le concept inconnu associé. Par exemple, les deux articles *Rouge* et *Jaune* sont classés sous le LCS *Bande Spectrale*. Il faut noter que WIKIPEDIA est utilisée par plusieurs approches pour l'extraction d'information sémantique.

```
 1: Routine COMPUTE-GENERAL-CONCEPTS
 2: Input : List 𝒞_d of derived concepts c_di
 3: Local : Binary matrix M = (𝒞_d × W), where W is a set of Wikipedia categories and m_ij ∈ M
    is set to 1 if the concept c_di is mapped to the category W_j
 4: Output : Ordered list 𝒞_g of general concepts c_gj
 5:
 6: BEGIN
 7: INITIALIZE-MATRIX(M)      {set elements to 0}
 8: for all c_di ∈ 𝒞_d do
 9:    W_i = GET-WIKIPEDIA-CATEGORIES(c_di)
10:    UPDATE-MATRIX(W,W_i)
11:    for all w_k ∈ W_i do
12:       SET-MATRIX(c_di,W_k,1)
13: 𝒞_g = GET-TOP-RATED-CATEGORIES(M)
14: Return 𝒞_g
15: END
```

Algorithm 4: Calcul des concepts généraux avec WIKIPEDIA.

L'algorithme 4 calcule les concepts généraux les plus pertinents d'un ensemble de termes (articles WIKIPEDIA). D'une manière générale, Pour tout terme en entrée, l'algorithme cherche les catégories WIKIPEDIA correspondante (ligne 9) ; ensuite, l'algorithme construit de façon progressive (ligne 10) une matrice d'incidence qui relie les termes en entrée à leurs catégories WIKIPEDIA respectives (ligne $11-12$). À la fin, l'algorithme détermine les concepts généraux les plus référencés par la matrice (ligne 13).

D'autre part, WORDNET semble être une ressource de choix pour la fouille dirigée par l'intension. En effet, beaucoup de propriétés ontologiques correspondent à des termes dans WORDNET. L'ontologie lexicale WORDNET est accessible, largement utilisée et développée et maintenue par la communauté de la recherche.

De plus, de point de vue de l'implémentation, aussi bien WIKIPEDIA que WORDNET offrent aux développeurs une API Java riche et efficace qui nous permettra d'implémenter avec aisance les les deux méthodes de fouille au sein de l'outil TOPICMINER qui sera au final intégré à la plateforme de génie ontologique INUKHUK.

4.4 CONCLUSION

La restructuration d'une ontologie est une forme d'enrichissement qui consiste parfois à introduire de nouvelles abstractions dont le but est de factoriser les propriétés de concepts existants. La restructuration pour laquelle on se propose de mettre en place une méthode de fouille de noms s'appuie sur RCA, un cadre formel qui produit deux types de descriptions pour toute nouvelle abstraction. La nouvelle abstraction, bien que dépourvue d'un nom de concept, possède deux descriptions. La première description représente les propriétés des concepts factorisés, tandis que la deuxième représente les noms de ces concepts.

Dans ce chapitre, nous avons décrit une approche de fouille de concepts à partir d'une brève description. Cette approche sera utilisée dans l'affectation de nom aux nouvelles abstractions crées par la méthode RCA dans le cadre de l'enrichissement d'une ontologie avec la plate-forme INUKHUK. L'approche proposée exploite des ressources linguistiques et ontologiques, notamment WORDNET et WIKIPEDIA.

CHAPITRE 5

EXPÉRIMENTATION

5.1 IMPLÉMENTATION DE TOPICMINER

Dans le cadre de ce travail, nous avons mis en place un utilitaire appelé TOPICMINER. TOPICMINER est une application '*standalone*' qui fait partie de la plateforme INUKHUK. Le but de TOPICMINER est d'annoter les concepts formels découverts par RCAENGINE. À la fin du processus de restructuration, ces concepts sont traduits en concepts ontologiques par OWLDESIGNER, un autre outil de la suite INUKHUK.

La Figure 5.2 illustre les fonctionnalités de base de TOPICMINER, à savoir :

- Fouille d'un nom de concept ontologique à partir de l'intension du concept formel correspondant,
- Fouille d'un nom de concept ontologique à partir de l'intension du concept formel correspondant,
- Fouille mixte qui consiste à confronter les résultats des deux fouilles ci-dessus et de trouver un nom plus approprié au concept ontologique candidat.

La Figure 5.3 montre quelques classes servant à l'implémentation de TOPICMINER. À chaque

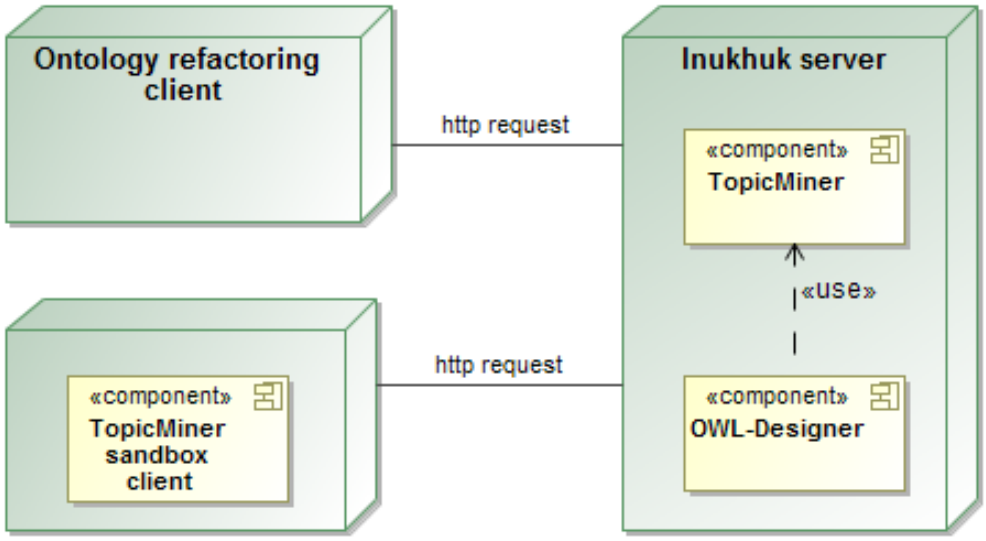


Figure 5.1: Le modèle de déploiement de TOPICMINER au sein de la plateforme INUKHUK.

méthode de fouille correspond un contrôleur, par exemple, INTENTBASEDMINER, EXTENDBASEDMINER, CROSSOVERMINER, etc. Les bases de connaissances sont aussi représentées et isolées de la logique métier en utilisant le patron de conception *Repository*. Une autre classe de type *Singleton* est utilisée pour fournir les distances sémantiques entre annotations en utilisant l'API WS4J.

5.2 PROTOCOLE DE VALIDATION

Afin de mener des tests de validation, nous avons mis en place le protocole de validation qui consiste à confronter les résultats retournés par TOPICTMINER à ceux fournis par un expert humain, en l'occurrence un concepteur d'ontologies.

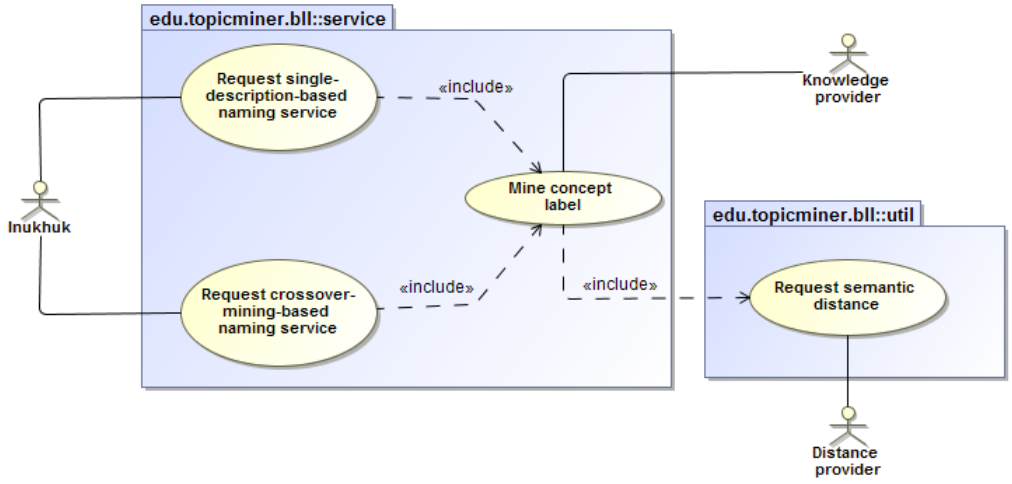


Figure 5.2: Les cas d'utilisation de TOPICMINER.

De plus, afin de juger les performances d'annotation de TOPICTMINER, nous allons calculer la distance sémantique entre le résultat fourni par TOPICTMINER et celui fourni par un concepteur d'ontologies. Un expert en restructuration d'ontologie est invité à juger la qualité du résultat d'annotation en se basant sur les différentes mesures de la distance sémantique.

D'autre part, les deux paramètres, la précision et le rappel, sont calculés. Pour ce faire, nous définissons les deux ensembles suivants :

— Ensemble N^c des noms n_i^c calculés par TOPICMINER.

— Ensemble N^r des noms n_j^r donnés par un expert humain ou un outil rival.

Le but est d'étudier la qualité des résultats de TOPICMINER en calculant les trois métriques classiques ci-dessous :

Précision : indique la proportion de bonnes valeurs parmi celles fournies par l'outil TOPICMINER.

$$P = \frac{\#Correct(n_i^c)}{|N^c|}$$

Rappel : indique la proportion de bonnes valeurs retournées par l'outil TOPICMINER

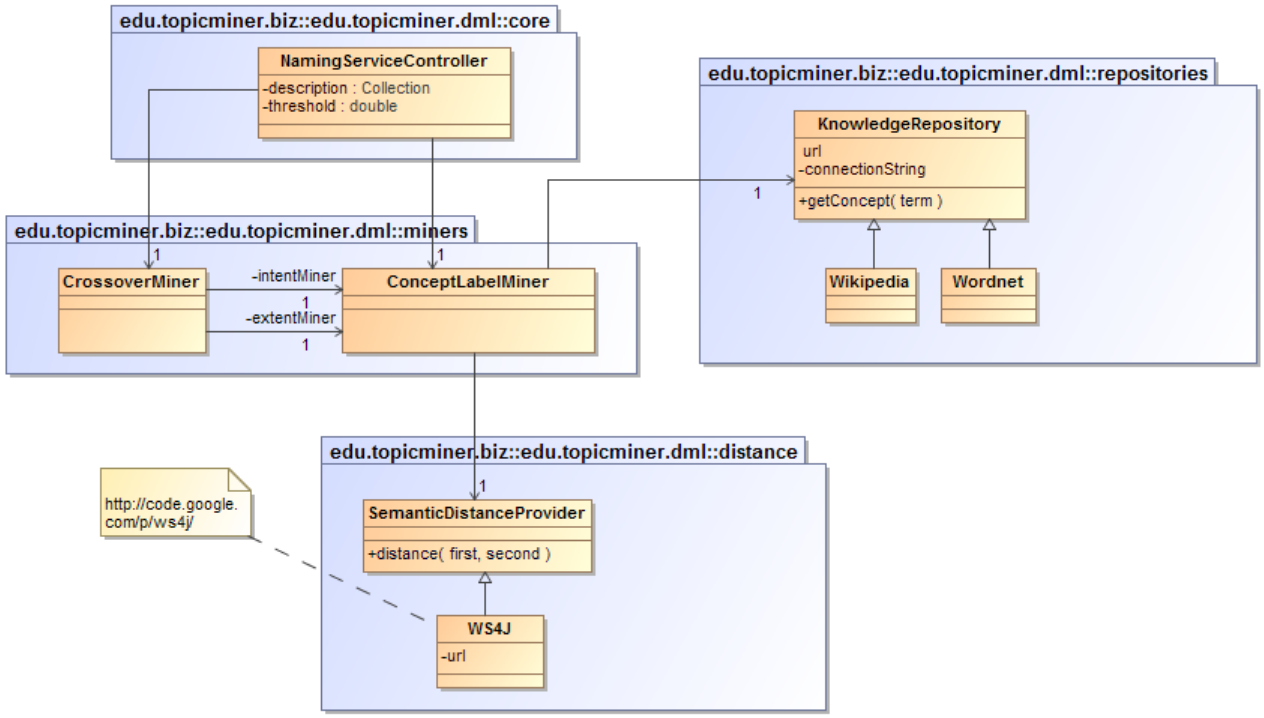


Figure 5.3: Le modèle de classes de conception de TOPICMINER.

parmi celles attendues.

$$R = \frac{\#Correct(n_i^c)}{|N^r|}$$

F-mesure : combinaison des deux métriques ci-dessus.

$$F = \frac{2 \times P \times R}{P + R}$$

Pour terminer la validation, notre protocole prévoit l'usage de TOPICMINER au sein de la plate-forme INUKHUK pour l'annotation des treillis de concepts formels produits par RCAENGINE. Une table de statistiques est produite récapitulant les résultats de nos expérimentations.

5.3 ÉVALUATION DES PERFORMANCES DE TOPICMINER

La table illustrée par la Figure 5.4 montre les expérimentations de la méthode de fouille de noms de concepts ontologiques à partir des extensions des concepts formels correspondants. La table donne les distances sémantiques entre l'annotation retournée par TOPICMINER et celle proposée par un concepteur d'ontologies. Plusieurs types de distances sémantiques sont mentionnés, tous calculés avec la populaire API WS4J.

• Mining Controller: **Extent Based Miner** • Learning Engine Workflow: **WPCAT_WNFIL_WPLCS** • Active Knowledge Base: **WIKIPEDIA** • Helper Resource: **WORDNET**				Semantic distance TopicMiner- Ontology Designer (WS4J API)					
	Formal Extend	TopicMiner Toolkit Annotation	Ontology Designer	JCN	LCH	RES	PATH	LESK	HSO
1	[Boy, Girl]	People	Person	00.14	01.9	02.87	00.17	583.00	05.00
2	[Dog, Wolf]	Canis_lupus	Animal	00.00	01.49	04.74	00.11	41.00	00.00
3	[Dog, Wolf, Cat]	Behavior	Animal	00.08	01.29	00.00	00.09	50.00	00.00
4	[Square, Rectangle]	Polygon	Figure	01.20	02.60	07.02	00.33	60.00	06.00
5	[NY, Boston]	Northeastern_US	City	00.13	02.08	04.67	00.20	127.00	03.00
6	[Earth, Mars]	Solar_System_objects	Planet	N/A	N/A	N/A	N/A	N/A	N/A
7	[Author, Reviewer]	Evaluation	Participant	00.06	01.20	00.00	00.08	48.00	00.00
8	[Paper, Report]	Communication	Submission	00.15	02.60	03.07	00.33	116.00	06.00

Figure 5.4: Fouille d'annotation à partir de l'extent. Distance sémantique entre les annotations de TOPICMINER et celles du concepteur de l'ontologie.

D'autre part, la table illustrée par la Figure 5.5 montre les expérimentations de la méthode de fouille de noms de concepts ontologiques à partir des extensions des concepts formels correspondants. Toutefois, la table donne les deux distances les plus significatives de la suite WS4J, à savoir :

WUP : cette mesure calcule l'analogie en considérant les profondeurs des deux synsets dans les taxonomies WordNet, ainsi que la profondeur du LCS.

LIN : cette mesure de distance s'appuie sur la notion de contenu de l'information (sous la forme de la probabilité conditionnelle de rencontrer une instance d'un enfant-synset à partir d'une instance d'un synset parent)

• Mining Controller: **Extent Based Miner** • Learning Engine Workflow: **WPCAT_WNFIL_WPLCS** • Active Knowledge Base: **WIKIPEDIA** • Helper Resource: **WORDNET**				Semantic distance TopicMiner- Ont. Design. (WS4J API)		
	Formal Extend	TopicMiner Annotation	Ontology Designer	WUP	LIN	Judge Rating
1	[Boy, Girl]	People	Person	00.62	00.33	100%
2	[Dog, Wolf]	Canis_lupus	Animal	00.67	00.00	100%
3	[Dog, Wolf, Cat]	Behavior	Animal	00.29	00.00	50%
4	[Square, Rectangle]	Polygon	Figure	00.86	00.94	100%
5	[NY, Boston]	Northeastern_US	City	00.78	00.54	90%
6	[Earth, Mars]	Solar_System_objects	Planet	N/A	N/A	100%
7	[Author, Reviewer]	Evaluation	Participant	00.26	00.00	40%
8	[Paper, Report]	Communication	Submission	00.80	00.48	100%

Figure 5.5: Fouille d'annotation à partir de l'extent. Distance sémantique entre les annotations de TOPICMINER et celles du concepteur de l'ontologie. De plus, la décision d'un expert.

La table illustrée par la Figure 5.5 illustre l'annotation retournée par TOPICMINER et celle proposée par un concepteur d'ontologies. Un jugement d'un expert en restructuration est fourni à partir des deux mesures WUP et LIN. On remarque que, dans beaucoup de cas (plus que 60% des cas), TOPICMINER arrive à annoter le concept formel de manière adéquate.

• Mining Controller: **Intent Based Miner** • Learning Engine Workflow: **WPCAT_WNFIL_WNLCS** • Active Knowledge Base: **WORDNET** • Helper Resource: **WIKIPEDIA**				Semantic distance TopicMiner- Ontology Designer (WS4J API)					
	Formal Extend	TopicMiner Annotation	Ontology Designer	JCN	LCH	RES	PATH	LESK	HSO
1	[Eat, Walk]	Achievement	Behavior	00.11	02.08	02.60	00.20	31.00	03.00
2	[Length, Width]	Dimension	Dimension	128.0	03.69	11.77	01.00	1998	16.00
3	[Run, Jump]	Long jump	Competition	00.00	02.30	07.16	00.25	25.00	05.00
4	[Hot, Cold]	Temperature	Climate	00.09	01.74	02.40	00.14	25.00	00.00
5	[Name, Email, Affiliate]	Communication	Participant	00.10	01.61	00.00	00.13	77.00	00.00
6	[Contents, Paper, Title]	Knowledge	Publication	00.11	02.08	01.78	00.20	159.00	03.00
7	[Compose, Write]	Structure	Submit	00.07	01.39	00.32	00.14	30.00	00.00
8	[Write, Rate]	Value	Formulate	00.12	01.72	03.62	00.20	32.00	03.00

Figure 5.6: Fouille d'annotation à partir de l'intent. Distance sémantique entre les annotations de TOPICMINER et celles du concepteur de l'ontologie.

La table illustrée par la Figure 5.6 montre les expérimentations de la méthode de fouille de noms de concepts ontologiques à partir des intensions des concepts formels correspondants. La table donne les distances sémantiques entre l'annotation retournée par TOPICMINER et

celle proposée par un concepteur d'ontologies. Les distances sémantiques calculées par l'API WS4J sont indiquées.

• Mining Controller: **Intent Based Miner** • Learning Engine Workflow: **WPCAT_WNFIL_WNLCS** • Active Knowledge Base: **WORDNET** • Helper Resource: **WIKIPEDIA**				Semantic distance TopicMiner- Ont. Design. (WS4J API)		
	Formal Extend	TopicMiner Annotation	Ontology Designer	WUP	LIN	Judge Rating
1	[Eat, Walk]	Achievement	Behavior	00.75	00.37	80%
2	[Length, Width]	Dimension	Dimension	01.00	01.00	100%
3	[Run, Jump]	Long jump	Competition	00.82	00.00	100%
4	[Hot, Cold]	Temperature	Climate	00.57	00.29	80%
5	[Name, Email, Affiliate]	Communication	Participant	00.36	00.00	45%
6	[Contents, Paper, Title]	Knowledge	Publication	00.60	00.28	90%
7	[Compose, Write]	Structure	Submit	00.36	00.40	40%
8	[Write, Rate]	Value	Formulate	00.60	00.47	70%

Figure 5.7: Fouille d'annotation à partir de l'intent. Distance sémantique entre les annotations de TOPICMINER et celles du concepteur de l'ontologie. De plus, la décision d'un expert.

D'autre part, la table illustrée par la Figure 5.7 montre les expérimentations de la méthode de fouille de noms de concepts ontologiques à partir toujours des intensions des concepts formels correspondants. Cette fois-ci, la table donne les deux distances WUP et LIN de la suite WS4J. De plus, la table montre le niveau d'adéquation fourni par un expert en restructuration à partir des deux mesures WUP et LIN. Là aussi, on remarque que notre outil arrive à annoter le concept formel de manière adéquate à partir de leurs intensions.

	Précision	Rappel	F-Mesure
Extent-Based-Miner	75%	66%	70%
Intent-Based-Miner	62%	50%	55%

Tableau 5.1: Précison et rappel des algorithmes de fouilles de TOPICMINER.

La Table 5.1 montre que la fouille à partir des extenstions des concepts formels est plus fructueuse. Ceci nous conduit à la conclusion que l'usage de WIKIPEDIA est plus avantageux que celui de WORDNET dans l'annotation des concepts formels dans le cadre de la génération d'une ontologie formels intelligible.

5.4 DÉPLOIEMENT DE TOPICMINER DANS LA PLATEFORME INUKHUK

5.4.1 EXEMPLE D'INTÉGRATION DE TOPICMINER À LA SUITE INUKHUK

À titre d'illustration et afin de simplifier les différents codages, nous avons opté pour une ontologie de petite taille. Étant donnée l'ontologie CMS illustrée par la Figure 5.8. Cette ontologie représente les connaissances d'un système de gestion de conférences. Ainsi, un auteur (le concept *Author*) peut soumettre (la relation sémantique *Wite*) plusieurs articles (le concept *Article*). Un évaluateur (le concept *Reviewer*) doit rédiger (la relation sémantique *Compose*) un rapport (le concept *Report*). Finalement, un rapport concerne (la relation sémantique *rate*) un papier soumis (le concept *Paper*).

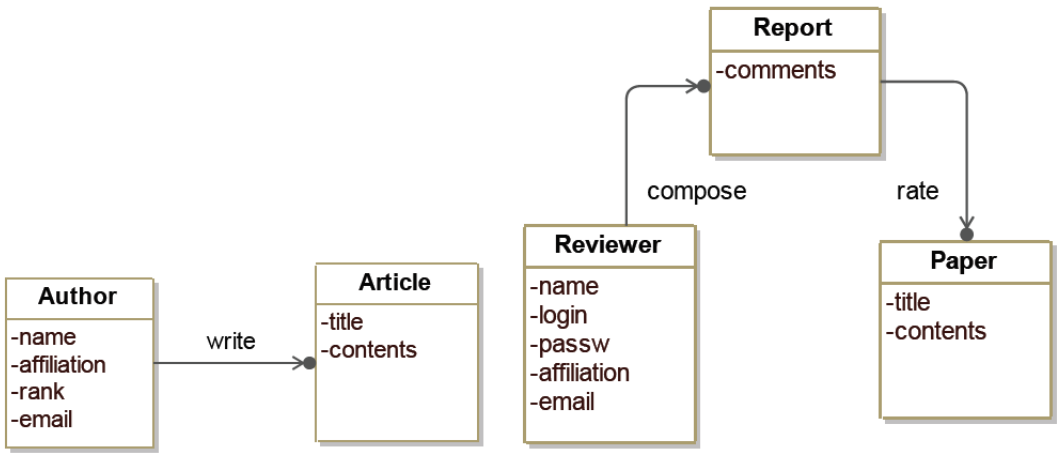


Figure 5.8: L'ontologie CMS. La version initiale.

Comme illustré dans la Figure 5.8, il est clair que cette ontologie présente des anomalies, d'une part, sous la forme d'une redondance des connaissances (par exemple, dans la description des deux concepts *Author* et *Reviewer*) et, d'autre part, l'absence de connaissances pertinentes au domaine du modèle, notamment la relation sémantique qui modélise les liens entre un participant (un *Author* ou un *Reviewer*) et une communication (*Article* et *Report*).

Le but ultime de la restructuration d'ontologies est de corriger ces anomalies. Pour ce faire, la plateforme INUKHUK, par le biais de RCFMODELER code l'ontologie initiale CMS en une famille de contextes formels tel que illustrée par la Figure 5.9. Ainsi, il y a deux contextes formels :

Contexte des concepts : code les concepts de l'ontologie initiale,

Contexte des relations : code les relations sémantiques de l'ontologie initiale.

Classes	author	reviewer	paper	report	name	affiliation	email	rank	title	contents	comments	login	passw
Author	×				×	×	×	×					
Paper			×						×	×			
Reviewer		×			×	×	×					×	×
Report				×							×		

Properties	'write'	'compose'	'rate'
write	×		
compose		×	
rate			×

Figure 5.9: Le codage des concepts et relations sémantiques de l'ontologie CMS en format RCF produit par l'outil RCFMODELER de la plateforme INUKHUK.

Le contexte des concepts montre les concepts ontologiques, les propriétés de ces concepts et la relation d'incidence Concept-Propriété. Le contexte des relations indique les relations sémantiques, leurs attributs et la relation d'incidence Relation-attribut.

La Figure 5.10 montre le codage des liens entre les concepts ontologiques et les relations sémantiques au sein d'une RCF.

source	write	compose	rate
Author	×		
Paper			
Reviewer		×	
Report			×

target	write	compose	rate
Author			
Paper	×		×
Reviewer			
Report		×	

dom	Author	Paper	Reviewer	Report
write	×			
compose			×	
rate				×

ran	Author	Paper	Reviewer	Report
write		×		
compose				×
rate		×		

Figure 5.10: Le codage des liens entre les concepts et les relations sémantiques dans l'ontologie CMS en format RCF produit par l'outil RCFMODELER de la plateforme INUKHUK.

L'outil RCAENGINE traduit la RCF en une famille de treillis (une RLF). La Figure 5.11 donne les deux ontologies formelles produites. À noter que l'approche RCA implémentée par la plateforme INUKHUK produit une ontologie formelle par contexte. Ainsi, nous avons une ontologie formelle pour les concepts ontologiques et une ontologie formelle des relations sémantiques de l'ontologie initiale CMS. Chaque ontologie formelle illustre toutes les abstractions potentielles dans le contexte correspondant.

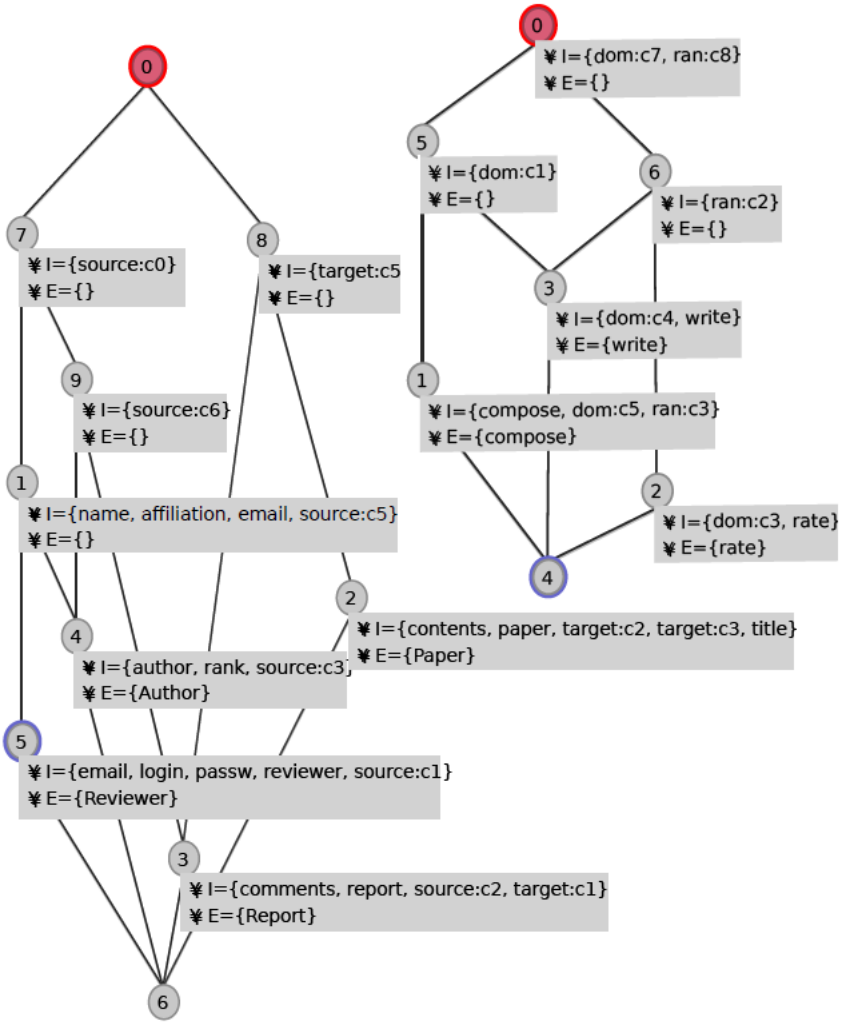


Figure 5.11: La famille de treillis de concepts produite par l'outil RCA-ENGINE de la plateforme INUKHUK. Cette RLF sera utilisée par l'outil ONTOLOGYDESIGN pour générer l'ontologie restructurée.

Les ontologies formelles (treillis de concepts dans le jargon de l'analyse formelle de concepts) fournissent à l'outil de génération de l'ontologie finale, dit OWLDESIGNER tous les candidats potentiels pour une meilleure restructuration de l'ontologie initiale. OWLDESIGNER utilise des filtres, notamment celui de la pertinence par rapport au domaine de l'ontologie, pour générer les concepts ontologiques de l'ontologie cible (restructurée). La Figure 5.12 illustre l'ontologie restructurée produite. Les concepts ontologiques *Evaluation*et *Communication*

ainsi que la relation sémantique *Value* ont tous été annotés par notre outil TOPICMINER.

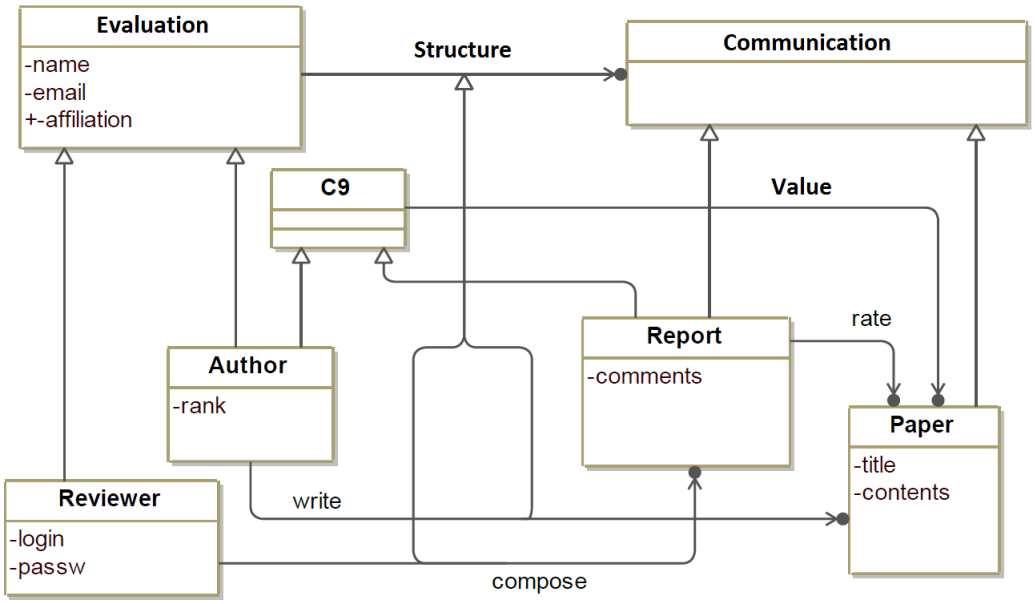


Figure 5.12: La version finale de l'ontologie CMS produite par ONTOLOGYDESIGNER et annotée par TOPICMINER.

Finalement, La Table 5.2 montre TOPICMINER en action au sein de la plateforme INUKHUK. On constate que TOPICMINER améliore sensiblement la qualité de la restructuration de la plateforme INUKHUK en annotant dans 80% des cas les concepts et relations découvertes. Par exemple, l'ontologie du tourisme contient 78 concepts et 26 relations. Après restructuration, 21 concepts nouveaux et 37 relations nouvelles ont été proposés dont 12 concepts et 15 relations ont trouvé une annotation avec l'outil TOPICMINER. Dans certaines situations (p.ex., le nommage des nouvelles relations de l'ontologie People), l'annotation est peu fructueuse.

	NCOI	NROI	NCOF	NROF	NCA	NRA	TAC	TAR
Tourism	78	26	99	63	12	37	57%	41%
Travel	35	06	38	10	03	02	100%	50%
People	60	14	64	16	03	00	75%	0%

Tableau 5.2: Performance de TOPICMINER au sein de la plateforme TOPICMINER. N : Nombre, C : Concepts, R : relations, O : Ontologie, I : Initial, F : Finale, A : Annotés. TAC : Taux d'annotation des concepts. TAR : Taux Annotation des relations.

CHAPITRE 6

CONCLUSION GÉNÉRALE

L'Analyse Relationnelle de Concepts (ARC) a été utilisée dans la réingénierie des ontologies Hacene et al. (2010) assurant ainsi une factorisation maximale des propriétés des concepts et la détection d'abstractions de concepts et rôles ontologiques potentiellement utiles. L'ARC a été implémenté au sein de la plateforme GALICIA. Une plateforme de ré-ingénierie d'ontologie, appelée INUKHUK a été mise en place et couplée avec GALICIA par le biais du module RCAENGINE pour le codage et construction des concepts formels qui sont par la suite traduits en concepts ontologiques. Toutefois, certains concepts ontologiques, nouveaux, ne possèdent pas de noms appropriés. En effet, ils portent les identifiants des concepts formels correspondants produits par RCAENGINE. Pour pallier à ce problème, nous avons proposé une nouvelle approche d'annotation permettant de rendre l'ontologie restructurée plus intelligible en affectant des noms appropriés aux nouveaux éléments (concepts et rôles ontologiques). Cette approche s'appuie sur les ressources linguistiques (WORDNET) et sur le Web (WIKIPEDIA). Nous avons aussi implémenté notre approche dans un nouveau outil, appelé TOPICMINER.

Finalement, nous avons intégré TOPICMINER à la plateforme INUKHUK et nous avons réalisé une série d'expérimentation pour analyser ses performances ainsi que son apport au processus de ré-ingénierie des ontologies. Bien que l'outil est en phase expérimentale, les résultats

obtenus sont encourageants. Plusieurs aspects de l'outil méritent d'être améliorés, tant sur le plan fondamental que celui de l'expérimentation, notamment la diversifications des bases de connaissances utilisées pour la fouille.

Afin d'améliorer la qualité des modèles ontologiques produits par TOPICMINER, nous continuons à travailler sur la performance de l'ensemble des algorithmes utilisés avec les différents types de fouilles ainsi que la recherche de nouveaux algorithmes permettant notamment de réduire les annotations fortuites.

BIBLIOGRAPHIE

Agrawal, R., T. Imielinski, et A. Swami. 1993. « Mining association rules between sets of items in large databases ». In *Proceedings of the ACM SIGMOD International Conference on the Management of Data, Washington (DC), USA*, p. 207–216.

Arnold, R. S. 1992. « Software Reengineering », *IEEE Computer Society Press*.

Auer, S., C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, et Z. Ives. 2007. « Dbpedia : A nucleus for a web of open data ». In *Proceedings of the 6th International The Semantic Web and 2Nd Asian Conference on Asian Semantic Web Conference*. Coll. « ISWC'07/ASWC'07 », p. 722–735, Berlin, Heidelberg. Springer-Verlag.

Barbut, M. et B. Monjardet. 1970. *Ordre et Classification : Algèbre et combinatoire*. Hachette.

Bennett, K. H. 1995. « Legacy systems : Coping with success », *IEEE Software*, vol. 12, no. 1, p. 19–23.

Birkhoff, G. 1940. *Lattice Theory*. T. XXV, série *AMS Colloquium Publications*. AMS.

Bordat, J. P. 1986. « Calcul pratique du treillis de Galois d'une correspondance », *Mathématique et Sciences Humaines*, no. 96, p. 31–47.

Carpineto, C. et G. Romano. 1996. « A Lattice Conceptual Clustering System and its Application to Browsing retrieval », vol. 24, no. 2, p. 95–122.

Cross, C. E. et J. II. 1990. « Software Reengineering », *Software, IEEE*, vol. 7, p. 13–17.

Dao, M., M. Huchard, M. R. Hacène, C. Roume, et P. Valtchev. 2004. « Improving Generalization Level in UML Models : Iterative Cross Generalization in Practice ». In *ICCS'04*, p. 346–360.

Davey, B. A. et H. A. Priestley. 1992. *Introduction to lattices and order*. Cambridge University Press.

Diday, E. et R. Emillion. 1997. « Treillis de galois maximaux et capacités de choquet », *C.R. Acad. Sci. Paris*, vol. 325, no. 1, p. 261–266.

Duquenne, V. 1999. « Lattice structures in data analysis », vol. 217, p. 407–436.

Egozi, O., E. Gabrilovich, et S. Markovitch. 2008. « Concept-based feature generation and selection for information retrieval ». In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*. Coll. « AAAI'08 », p. 1132–1137. AAAI Press.

Fowler, M., K. Beck, J. Brant, W. Opdyke, et D. Roberts. 1999. *Refactoring : Improving the Design of Existing Code*. Addison-Wesley. Object Technologies Series.

Gabrilovich, E. et S. Markovitch. 2005. « Feature generation for text categorization using world knowledge ». In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*. Coll. « IJCAI'05 », p. 1048–1053, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Ganter, B. 1984. Two basic algorithms in concept analysis. preprint no. 831, Technische Hochschule, Darmstadt.

Ganter, B., J. Stahl, et R. Wille. 1986. « Measurement and many-valued contexts », *W. Gaul, M. Schader : Classification as a tool of research*, p. 169–176.

Ganter, B. et R. Wille. 1999. *Formal concept analysis - mathematical foundations*. Springer.

———. 2001. *Formal Concept Analysis. Mathematical Foundations*. Springer, Berlin.

Girard, R. 1997. « Classification conceptuelle sur des données arborescentes et imprécises ». Thèse de doctorat, Université de la Réunion.

Godin, R., H. Mili, G. Mineau, et R. Missaoui. 1995. « Conceptual clustering methods based on Galois lattices and applications », vol. 9, no. 2, p. 105–137.

Godin, R., H. Mili, G. Mineau, R. Missaoui, A. Arfi, et T. Chau. 1998. « Design of Class Hierarchies Based on Concept (Galois) Lattices », *Theory and Practice of Object Systems*, vol. 4, no. 2.

Godin, R. et P. Valtchev. 2003. « Formal concept analysis-based normal forms for class hierarchy design in oo software development ». In *FCA : State of the Art*. T. to appear. Springer.

Gruber, T. R. 1995. « Toward principles for the design of ontologies used for knowledge sharing », *Int. J. Hum.-Comput. Stud.*, vol. 43, no. 5-6, p. 907–928.

Guarino, N. 1998. *Formal Ontology in Information Systems : Proceedings of the 1st International Conference June 6-8, 1998, Trento, Italy*. Amsterdam, The Netherlands, The Netherlands : IOS Press, 1st édition.

Hacene, M. R., S. Fennouh, R. Nkambou, et P. Valtchev. 2010. « Refactoring of ontologies : Improving the design of ontological models with concept analysis ». In *ICTAI (2)*, p. 167–172. IEEE Computer Society.

Hacene, M. R., M. Huchard, A. Napoli, et P. Valtchev. 2013. « Relational concept analysis : mining concept lattices from multi-relational data », *Ann. Math. Artif. Intell.*, vol. 67, no. 1, p. 81–108.

Hepp, M., K. Siorpaes, et D. Bachlechner. 2007. « Harvesting wiki consensus : Using wikipedia entries as vocabulary for knowledge management », *IEEE Internet Computing*, vol. 11, no. 5, p. 54–65.

Huang, A., D. Milne, E. Frank, et I. H. Witten. 2009. « Clustering documents using a wikipedia-based concept representation ». In *Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*. Coll. « PAKDD '09 », p. 628–636, Berlin, Heidelberg. Springer-Verlag.

Huchard, M., H. Dicky, et H. Leblanc. 2000. « Galois lattice as a framework to specify algorithms building class hierarchies », *Theoretical Informatics and Applications*, vol. 34, p. 521–548.

Huchard, M., C. Roume, et P. Valtchev. 2002. « When concepts point at other concepts : the case of uml diagram reconstruction ». In *In Proceedings of the Workshop FCAKDD*, p. 32–43.

K. H. Bennett, B. J. Cornelius, M. M. et D. J. Robson. 1991. *Software Maintenance*. Butterworth Heinemann.

Kent, R. 1996. « Rough concept analysis : A synthesis of rough sets and formal concept analysis », *Funadamenta Informaticae*, vol. 27, p. 169–181.

Kuznetsov, S. et S. Ob'edkov. 2000. Algorithms for the Construction of the Set of All Concept and Their Line Diagram. preprint no. MATH-AL-05-2000, Technische Universität, Dresden.

Lehman, M. M. et L. A. Belady. 1985. *Program evolution*. Academic Press, New York.

Miller, G. A. 1995. « Wordnet : A lexical database for english », *Commun. ACM*, vol. 38, no. 11, p. 39–41.

Milne, D. et I. H. Witten. 2008. « Learning to link with wikipedia ». In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. Coll. « CIKM '08 », p. 509–518, New York, NY, USA. ACM.

Nehme, K., P. Valtchev, M. H. Rouane, et R. Godin. 2005. « On computing the minimal generator family for concept lattices and icebergs ». In Ganter, B. et R. Godin, éditeurs, *Proceedings of the 3rd Intl. Conference on FCA, Lens (FR), February 14-18*. Coll. « LNCS 3403 », p. 192–207. Springer Verlag.

Nourine, L. et O. Raynaud. 1999. « A Fast Algorithm for Building Lattices », *Information Processing Letters*, vol. 71, p. 199–204.

Piateski, G. et W. Frawley. 1991. *Knowledge Discovery in Databases*. Cambridge, MA, USA : MIT Press.

Prediger, S. 1997. « Logical scaling in formal concept analysis ». In *International Conference on Conceptual Structures*, p. 332–341.

Rahm, E. et P. A. Bernstein. 2001. « A survey of approaches to automatic schema matching », *VLDB Journal : Very Large Data Bases*, vol. 10, no. 4, p. 334–350.

Rouane, M. H. 2003. « Magalice : An incremental algorithm for iceberg lattice maintenance ». In *Proceedings of the Treillis de Galois et IA Workshop, Plateforme AFIA, Laval (FR), 4 July*, p. 77–86.

Snelting, G. 2000. « Software reengineering based on concept lattices ». In Society, I. C., éditeur, *Proc. of CSMR'00*.

Song, Y., H. Wang, Z. Wang, H. Li, et W. Chen. 2011. « Short text conceptualization using a probabilistic knowledgebase ». In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*. Coll. « IJCAI'11 », p. 2330–2336. AAAI Press.

Stankovic, M., W. Breitfuss, et P. Laublet. 2011. « Discovering relevant topics using dbpedia : Providing non-obvious recommendations ». In *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Volume 01*. Coll. « WI-IAT '11 », p. 219–222, Washington, DC, USA. IEEE Computer Society.

Stumme, G., R. Taouil, Y. Bastide, N. Pasquier, et L. Lakhal. 2002. « Computing Iceberg Concept Lattices with Titanic », *Journal on Knowledge and Data Engineering*, vol. 42, no. 2, p. 189–222.

Suchanek, F. M., G. Kasneci, et G. Weikum. 2007. « Yago : A core of semantic knowledge ». In *Proceedings of the 16th International Conference on World Wide Web*. Coll. « WWW '07 », p. 697–706, New York, NY, USA. ACM.

Tonella, P. et G. Antoniol. 1999. « Object oriented design pattern inference ». In *Proceedings of ICSM '99*, p. 230, Washington, DC, USA. IEEE Computer Society.

Valtchev, P., M. R. Hacene, et R. Missaoui. 2003a. « A generic scheme for the design of efficient on-line algorithms for lattices ». In de Moor, A., W. Lex, et B. Ganter, éditeurs, *Proceedings of the 11th Intl. Conference on Conceptual Structures (ICCS'03)*. T. 2746, série *Lecture Notes in Computer Science*, p. 282–295, Berlin (DE). Springer-Verlag.

Valtchev, P. et R. Missaoui. 2000. « Exploration of Complex Objects Structure for Knowledge Discovery ». In *Workshop on Structured Data in Machine Learning and Statistics, ECML'2000 (to appear)*.

Valtchev, P., R. Missaoui, et R. Godin. 2002. « A framework for incremental generation of frequent closed itemsets ». In *Proceedings of the 1st International Workshop on Discrete Mathematics and Data Mining*, Washington (DC), USA.

Valtchev, P., M. H. Rouane, D. Grosser, et C. Roume. 2003b. « An open platform for manipulating lattices ». In *Proceedings of the 11th ICCS, Dresden, Germany*.

Vogt, F. et R. Wille. 1994. « TOSCANA – a graphical tool for analyzing and exploring data ». In Tamassia, R. et I. G. Tollis, éditeurs, *Graph Drawing*. T. 894, p. 226–233.

Wille, R. 1982. « Restructuring the lattice theory : an approach based on hierarchies of concepts », *Ordered Sets*, p. 445–470.